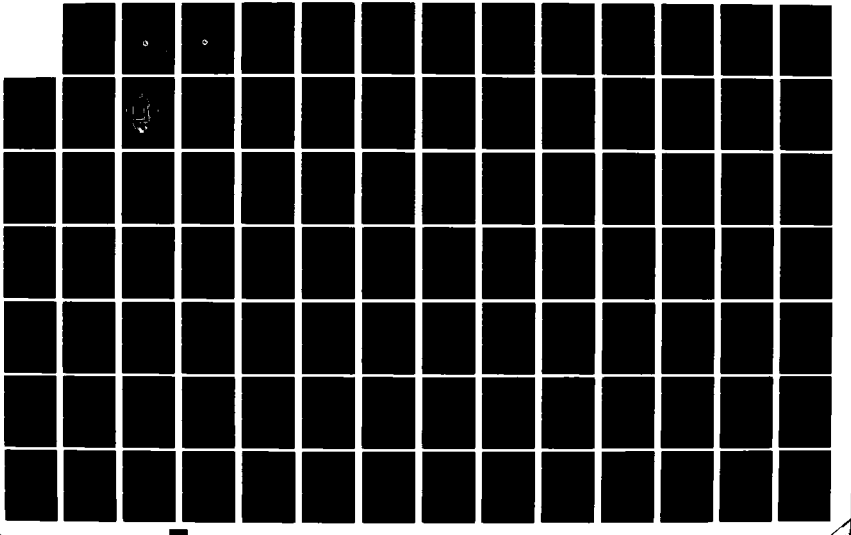
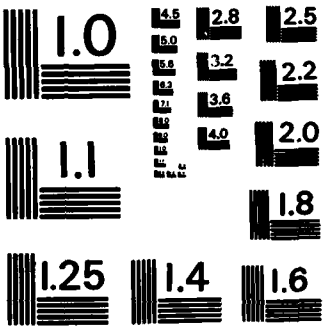
AD-A121 759 PILOT: A PRECISION INTERCOASTAL LORAN TRANSLOCATOR VOLUME 3 SOFTWARE(U) JOHNS HOPKINS UNIV LAUREL MD APPLIED PHYSICS LAB G E BAER ET AL. MAR 82 1/2

UNCLASSIFIED USCG-D-21-81-3 N00024-78-C-5384 F/G 9/2 NL





Report No. CG-D-21-81, III

(12)

PILOT
A PRECISION INTERCOASTAL LORAN TRANSLOCATOR
VOLUME 3-SOFTWARE

GLEN E. BAER

JAMES F. HARRISON, JR.

of

The Johns Hopkins University Applied Physics Laboratory
Johns Hopkins Road, Laurel, Maryland 20707

March 1982

Final Report

Document is available to the U.S. public through the National Technical Information Service, Springfield, Virginia 22161

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
UNITED STATES COAST GUARD
Office of Research and Development
Washington, D.C. 20590

Report No. CG-D-21-81, III

PILOT
A PRECISION INTERCOASTAL LORAN TRANSLOCATOR
VOLUME 3-SOFTWARE

GLEN E. BAER

JAMES F. HARRISON, JR.

of

The Johns Hopkins University Applied Physics Laboratory
Johns Hopkins Road, Laurel, Maryland 20707

March 1982

Final Report

Document is available to the U.S. public through the
National Technical Information Service,
Springfield, Virginia 22161

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
UNITED STATES COAST GUARD
Office of Research and Development
Washington, D.C. 20590

NOTICE

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the object of this report.

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
CG-D-21-81, III	AD-A121759	

4. Title and Subtitle	5. Report Date
PILOT A Precision Intercoastal Loran Translocator	March 1982
	6. Performing Organization Code
	8. Performing Organization Report No.

7. Author(s)	
Glen E. Baer and James F. Harrison, Jr.	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
The Johns Hopkins University Applied Physics Laboratory Johns Hopkins Road Laurel, Maryland 20707	11. Contract or Grant No. N00024-78-C-5384
	13. Type of Report and Period Covered Final Report
12. Sponsoring Agency Name and Address United States Coast Guard Headquarters Washington, D.C. 20590	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This report discusses the software involved in a Coast Guard aid to navigation which evolved from the Loran Assist Device (LAD) to the development of microprocessors into the present PILOT system. The object of the PILOT system is to demonstrate that Loran repeatability can be used successfully to pilot harbors and rivers without significantly increasing the workload of bridge personnel.

17. Key Words	18. Distribution Statement
Loran Piloting Microprocessor PILOT Terminal	No restriction on distribution. Available from National Technical Information Service, Springfield, VA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
UNCLASSIFIED	UNCLASSIFIED		

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

PREFACE

a. The present PILOT terminal is the product of a long evolution of special purpose loran processors developed at the Applied Physics Laboratory. In 1968 a Loran Assist Device (LAD) was developed for a unique military aircraft requirement and was followed by several other military versions. In 1970 a U.S. Coast Guard Loran Assist Device (COGLAD) was developed to evaluate loran as an aid to positioning buoys. With the development of microprocessors in 1973, a small, simple processor (CLAD) was developed and tested by the U.S. Coast Guard. The original COGLAD was upgraded and tested on the St Marys River in 1976. Each new system utilized increasingly sophisticated data processing techniques, required less operator training and attention, and represented a lower potential production cost. These improvements were largely the result of the phenomenal developments in the integrated circuit and microprocessor industry in the last decade.

b. The design objective of the PILOT system was to demonstrate that loran repeatability (i.e., returning to presurveyed way points) could be used successfully to pilot harbors and rivers without significantly increasing the workload of the bridge personnel, and that the system could be mass produced by industry at an affordable price. To minimize development and production costs, a commercially available microprogrammable graphics terminal was selected as the nucleus of the PILOT system. Significant hardware modifications were required, but most are of the plug-in or bolt-on type and do not change the basic graphics terminal. The loran receiver currently used with the system is an unmodified commercial item. Extensive new software was developed for this application and included sophisticated data filtering and transformation techniques.

c. This report, documenting the PILOT system, is in three volumes: Users Manual (Vol. I), Hardware (Vol. II), and Software (this volume). Additional documentation is contained in three Hewlett-Packard publications.*

*Hewlett-Packard, Reference Manual for 2648A Graphics Terminal, Part No. 02648-90002, Hewlett-Packard Company, 19400 Homestead Road, Cupertino, California, 95014, 1978.

Hewlett-Packard, Technical Information Package for 264XX Data Terminal Part No 13255-91000, Hewlett-Packard Company, 19400 Homestead Road, Cupertino, California, 95014, 1977.

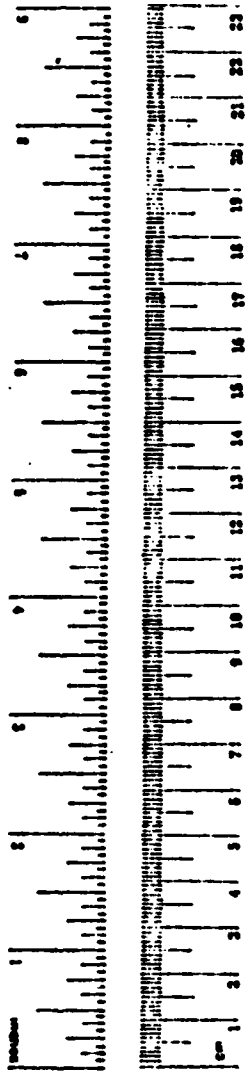
Hewlett-Packard, 2645A Operating System Microcode, Part No. 13255-90003, Hewlett-Packard Company, 19400 Homestead Road, Cupertino, California, 95014, 1976.

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.5	kilometers	km
	AREA			
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
	MASS (weight)			
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
	VOLUME			
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
	TEMPERATURE (exact)			
	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.25, SD Catalog No. C13.10:286.



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
	AREA			
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.6	acres	
	MASS (weight)			
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
	VOLUME			
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
	TEMPERATURE (exact)			
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	

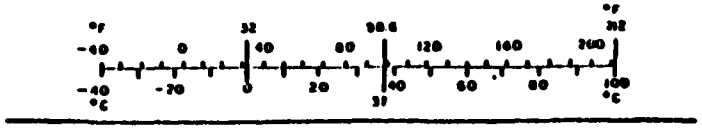


TABLE OF CONTENTS

CONTENTS (Continued)

CONTENTS (Concluded)

ILLUSTRATIONS

ILLUSTRATIONS (Concluded)

TABLES

ABBREVIATIONS

(Appearing on PILOT display and used throughout this manual.)

ATD	Along-Track Distance
ATS	Along-Track Speed
CTD	Cross-Track Distance
CTS	Cross-Track Speed
TTG	Time To Go

(Used throughout this manual.)

AMD	Advanced Micro Devices
APU	Arithmetic Processing Unit
ASCII	American Standard Code for Information Interchange
CMG	Course Made Good
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DR	Dead Reckoning
H	Hexadecimal
HP	Hewlett-Packard
I/O	Input-Output
LSB	Least Significant Byte
MSB	Most Significant Byte
OEM	Original Equipment Manufacturer
RAM	Random Access Memory
ROM	Read Only Memory
TC	Tape Convert
TD	Time Difference
WP	Waypoint

1.0 INTRODUCTION.

1.1 General:

a. The PILOT terminal, developed by the Applied Physics Laboratory of The Johns Hopkins University for the U.S. Coast Guard, is an electronic aid to piloting vessels in harbors and rivers. Its purpose is to provide accurate navigation information in a format that can be immediately used without significantly increasing the bridge workload. It is not intended to make value judgments or steer the ship, but it does provide the ship with continuous and accurate position information over-the-bottom. The PILOT terminal obtains its information from a Loran-C receiver, the ship's gyro, and charts and loran coefficients prerecorded on magnetic tape cartridges. These data are edited for accuracy, filtered for smoothness, and mathematically transformed into various display formats. The data are presented graphically with respect to a local way point (WP), and as a horizontal bar graph to aid channel keeping. The startup procedure for the PILOT system consists basically of selecting the appropriate tape cartridge and, after the receiver has automatically acquired the loran signals, commanding the PILOT terminal to start navigation. Once started, the PILOT system will continuously compute position and velocity information, and select new area charts as necessary without further operator action. The operator may enable optional features and display enhancements if desired.

b. Prerecorded tape cartridges containing a sequence of charts and other navigation information provide the PILOT terminal with a degree of "local knowledge". The vessel's present position, speed, and heading, continuously determined from a loran receiver and the vessel's gyro, are displayed on the current area chart. Charts of two different scales (master and detail) are always available for operator selection. Position, speed, and time to go (TTG) relative to way points are displayed to the left of the chart. A horizontal bar graph, representing the vessel's crosstrack position relative to the track line, can be displayed along the bottom of the display.

c. Other features available to the operator include a capability for projecting the vessel's expected track line, continuous readout of range and bearing from ownship to any point of the displayed chart, a time-of-day clock, and zoom in, zoom out, and pan on the display chart. The operator can enter time difference (TD) bias values to compensate for seasonal variation, select from three types of vessel projections, select from four data filter time constants, and display ship's gyro and loran receiver data. Using a special diagostic tape cartridge the operator can perform self tests on the PILOT terminal. When used with a line printer the PILOT terminal can print position information at a fixed time interval, or be used as a survey system to measure the mean and standard deviation of the TD's of the vessel's present position.

1.2 System Description:

a. The PILOT terminal is shown in Fig. 1-1, and the system block diagram is shown in Fig. 1-2. Normal shipboard installation consists of the PILOT terminal, a loran receiver (currently the Internav 404 receiver) and a cable connection to the ship's gyro. Optional equipment includes a second loran receiver (for cross chain operation), an interface or modem for remotely entering TD bias values, and a printer.

b. The nucleus of the PILOT system is an HP 2649A microprogrammable graphics terminal. This OEM device was selected because it has a separate graphics processor with memory, dual tape cartridge units, and an Intel 8080 microprocessor main processor, that could be modified and programmed as required. Modifications to the HP 2649A included: converting the 8080 microprocessor from software arithmetic to hardware arithmetic by adding an Advanced Micro Devices (AMD) 9511 arithmetic processing unit, developing an interface board to preprocess loran data from two loran receivers, receiver interface board, developing an interface board to connect to the ship's gyro, and a TD bias modem or box, replacing the large general purpose keyboard with a small predefined keypad, and building a short base for the terminal to serve as a cable junction box. The receiver interface board contains an Intel 8085 microprocessor and an AMD 9511 arithmetic processing unit.

1.3 Harbor Chart Tape Cartridge Generation:

a. All coordinate conversion constants and navigational reference information, as well as a mosaic of harbor charts, are contained on a cartridge which is, in effect, a Loran-C harbor chart on a magnetic medium. A single cartridge contains as many as 200 of these charts, representing about 200 miles of river or harbor.

b. Each cartridge contains an index file, master files, and detail files. The index file contains a title block and a list of all master charts on the cartridge. The index file is always displayed before the PILOT terminal begins the navigation mode and any time the terminal is reset.

c. Each master file contains the graphics for a master chart showing 8-16 miles of track (a scale of about 1:50,000), the area transformation coefficients, transmitter coordinates, and supplemental data such as display origin, scale, and rotation. Master charts provide "look ahead" by showing the next several way points and the identity of a few prominent features. Because master charts are generally not large enough for piloting in narrow channels, each master chart has one or more detail charts associated with it.

d. Each detail file contains the graphics for a detail chart showing 1-2 miles of track (a scale of about 1:12,500), the TD's, and X,Y coordinates of the current way point, bearing angles to and from the WP, and supplemental data. Detail charts provide a closer view of the vessel's current situation, and include many of the fixed aids to navigation and the outline of the channel if applicable.

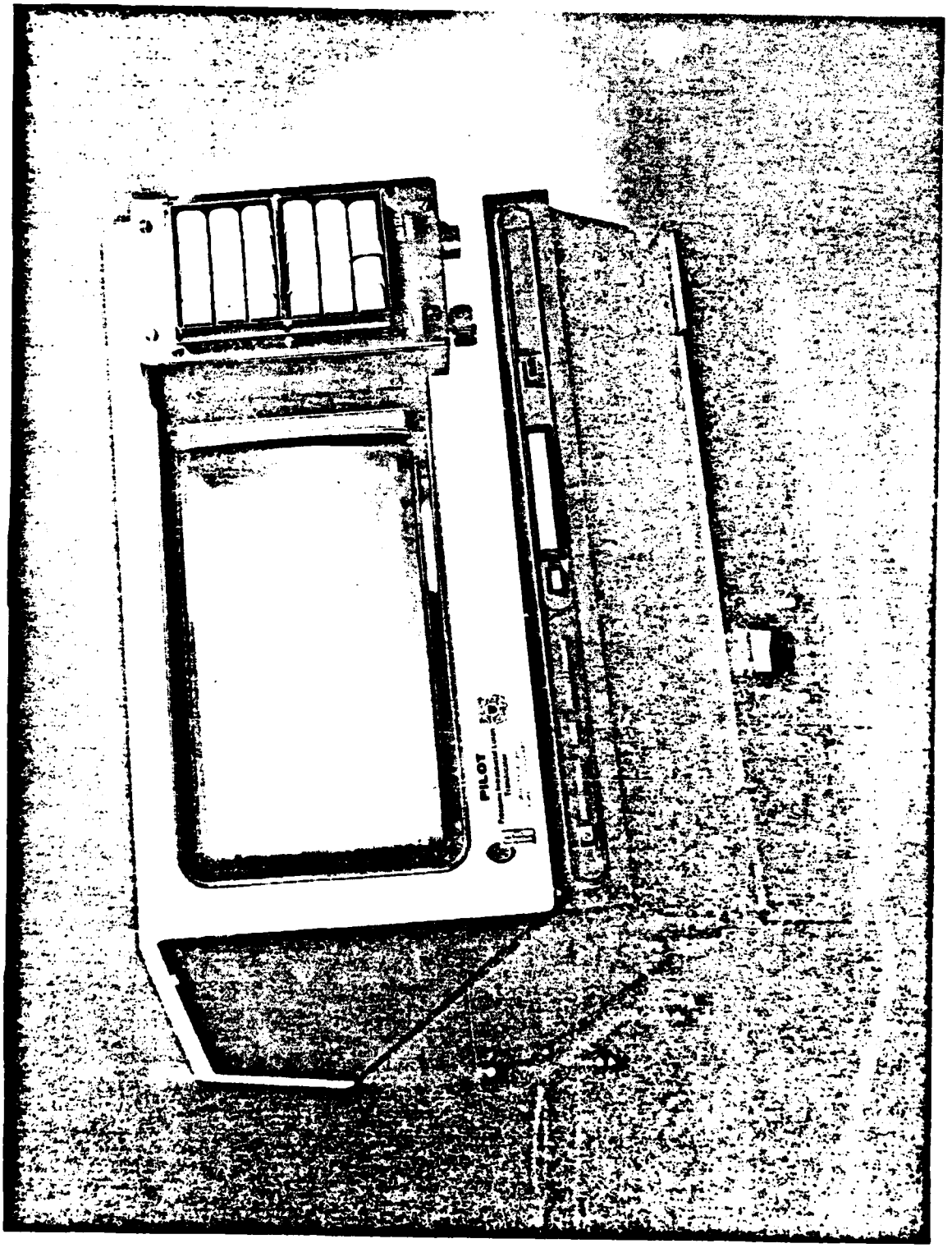


FIGURE 1-1. PILOT TERMINAL.

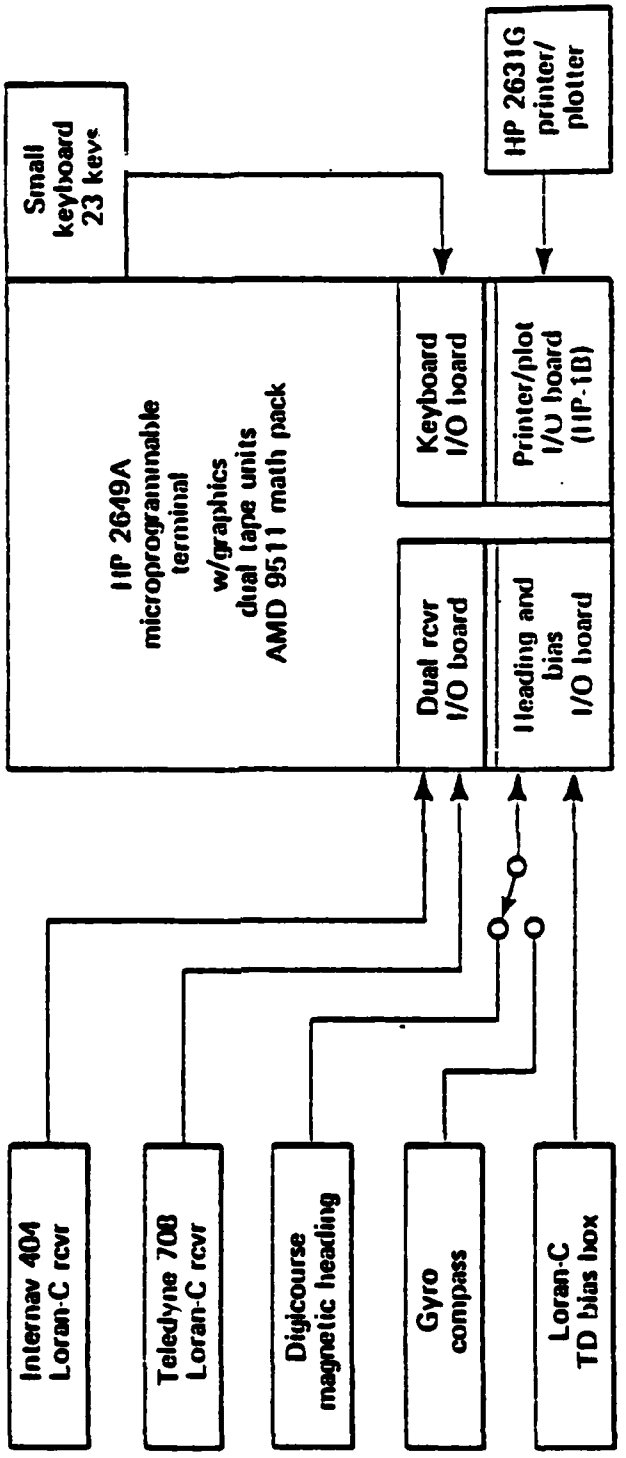


FIGURE 1-2. SYSTEM BLOCK DIAGRAM.

e. The tape convert software was developed to generate these cartridges. The tape convert software replaces the PILOT main code software. The procedure for generating these tapes is an off-line procedure.

f. The tape convert software requires a tape containing the information in ASCII, which enables the operator to change any or all files before generating the on-line or object tape which is used with PILOT.

2.0 SCOPE.

a. This PILOT software design manual provides a description of the software associated with the two microprocessors mentioned in Sections 1.2 and 1.3. The APL-developed code will be given considerably more coverage than Hewlett-Packard software, which is described in greater detail in Ref. 1.

b. The description of the APL-developed software is grouped in the following ways:

1. Main Processor Software (Intel 8080 microprocessor).

2. Receiver Interface Software (Intel 8085 microprocessor).

3. Tape Convert Software (Intel 8080 microprocessor for main processor).

c. Code for the Intel 8080 and 8085 microprocessors was written in the Intel 8080/8085 assembly language. Descriptions of the instruction sets associated with these languages are contained in Ref. 6

d. Development of the code was performed on an Intel MDS-230 Logic Development System. The operation of the assembler/linker and emulator/analyzer parts of this system is described in Refs. 3, 4, and 5 respectively.

e. The PILOT program developed software for two different microprocessors. The software associated with the main processor in the HP terminal is the PILOT software. Another set of software, Tape Convert, was developed for the generation of tapes to be used with the PILOT software. Tape Convert was also written for the main microprocessor in the HP terminal. The software associated with the Receiver Interface card is referred to as the REXEC software. These three sets of software are completely independent.

2.1 APL Developed Software:

2.1.1 Main Processor Software:

The main processor software or PILOT software receives time differences (TD's) from the REXEC software to compute the vessel's present position and speed. Inputs from the vessel's gyro and from the operator via the keyboard provide additional data for the PILOT software. This software is documented in Section 3.0.

2.1.2 Tape Convert Software:

The Tape Convert software transforms ASCII data into object data for use during real-time navigation by the PILOT software. This software is documented in Section 6.0.

2.1.3 REXEC Software:

The REXEC software communicates with at most two receivers and the PILOT software. The REXEC software was written to communicate with the Teledyne 708 and Internav Mark III Loran-C receiver combination. The REXEC REV A software was written to communicate with the Teledyne 708 and Internav LC404 Loran-C receiver combination. These sets of software are documented in Section 4.0.

3.0 PILOT MAIN PROCESSOR SOFTWARE.

3.1 PILOT Software:

a. The PILOT unit is developed around a Hewlett-Packard 2649A computer terminal. This terminal has all the basic parts required for a computer terminal, i.e., keyboard, display, and dual tape drives, plus an extensive graphics hardware and software package. The PILOT unit employs all these features in performing the navigation function. The HP 2649A terminal used is the OEM version of the HP 2648 graphics terminal. The basic operation of the terminal is described in Ref. 2. The software which was purchased with the system is documented in Ref. 1.

b. In developing the PILOT software, the HP software was used to communicate with all I/O devices. The PILOT software was appended to the existing HP software.

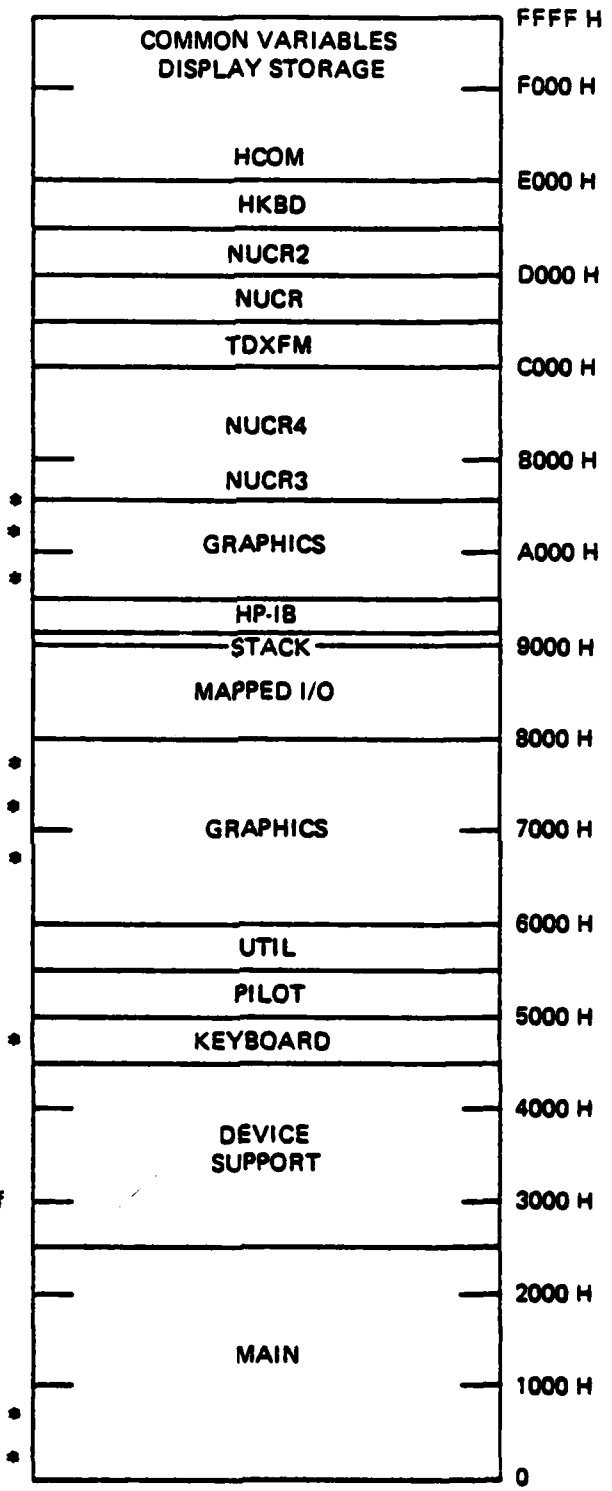
c. As documented in Ref. 1, the HP software initializes the terminal in anticipation of operator entries and then remains in a wait-loop until the operator takes some action on one of the I/O devices. The PILOT software permits the HP software to initialize the terminal. When the HP software enters the wait-loop, control is shifted permanently to the PILOT software. This is accomplished by modifying the HP code.

d. Although the PILOT software is in control of the terminal, the hardware interrupts have not been altered. As a result, the HP code is in control of the interrupts and interrupt structure.

3.1.1 PILOT Memory Organization:

a. The HP 2649A computer terminal has 64K bytes of addressable memory space. The PILOT program has divided this memory into 8K of RAM and 56K of EPROM. Of the 56K bytes of EPROM, 40K are HP code and 16K are PILOT code (see Fig. 3-1). The PILOT code has been divided into eight 2K blocks. The eight PILOT blocks are PILOT, UTIL, NUCR3, NUCR4, TDXFM, NUCR, NUCR2, and HKBD.

b. Each 2K block of PILOT software represents an independent set of software. All 2K blocks have the ability to use any or all of the variables located in the common variable area, HCOM. Each 2K block of software is linked independently of all other blocks. The entry vector technique is used to reference software not contained within the same block (see Section 3.1.3).



*Denotes 2K sections of Hewlett-Packard code which have minor modifications.

FIGURE 3-1. PILOT MEMORY MAP.

3.1.2 PILOT Software Structure:

a. The PILOT software is structured so that the executive routine, EXEC, is the main program. EXEC is the routine to which the HP code jumps after setting up the terminal and it remains there permanently.

b. Although no hardware interrupts have been altered, the routine which services the keyboard interrupt has been changed to permit the PILOT keyboard routine, HKDEC, to be called. HKDEC sets flags to indicate what action is to be taken by the PILOT executive.

c. The keyboard used in the PILOT unit is a smaller version of the HP keyboard. All HP keyboard codes were used in the final PILOT system. To service keys during operation a keyboard monitor routine, HKBD, is used. Calls to this routine are made throughout the PILOT executive to service keyboard entries.

d. The HP graphics package requires time to update the graphic display in order to handle all the graphics options. The keyboard monitor also performs this function.

3.1.3 PILOT Entry Vectors:

a. A segment of memory at the beginning of each 2K block is reserved for the definition of an absolute address of a routine which is used outside that block. The entry vector technique is used to reference software not contained within the same block.

b. Two routines exist for each 2K block. One defines the entry vectors within the block and the other provides a definition that other blocks can use. To use a routine outside a block, link the entry vector definition to satisfy the external reference. When the code is executed, the entry vector is transparent to the operation of the software.

c. Table 3-1 lists those routines in each 2K block which provide entry vector definitions, entry point definitions, and links to routines in the indicated blocks of code.

3.2 Hewlett-Packard Code Modifications:

3.2.1 Hewlett-Packard 2648 Graphics Code Without Autoplot:

a. The HP 2649A terminal used on the PILOT program has an extensive graphics package. The autoplot feature of the graphics package is not required. A version of the code without this feature was obtained from HP.

TABLE 3-1. ENTRY VECTOR AND ENTRY POINT DEFINITIONS AND LINKS.

ROUTINE (2K Block)	Entry Vector Definition	Entry Point Definitions and Links.
PTEV	PILOT	EXEC, TDFC, TDBOUT, MDRT, TDD, ESP, EFER, EXLOOP, and MDRT.
PTEVD	PILOT	PTEVD is linked to any block using routines in the PILOT block.
UEV	UTIL	LDMP1, LDMP2, STMP, MDRDY, ZRMP, DEGRAD, RADDEG, ESCI, ESCS, SQRX, FXSQRX, XFER, ASCFP, HYPON1, HYPON2, ATAN1, ATAN2, TRIG, EREX, DFASC, ASCDF, RWCRT, CRTRW, DEGRDN, ASCSF, STMP2, NEWWPT, and WPXY.
UEVD	UTIL	UEVD is linked to any block using routines in the UTIL block.
NU3EV	NUCR3	BGON, BGOFF, BLON1, BLON2, HDGFIL, GCP, IRAB, RAB, ERAB, IRABL, STAT, VESDIM, PROJ, PROJI, PROJX, and EREXP.
NU3EVD	NUCR3	NU3EVD is linked to any block using routines in the NUCR3 block.
NU4EV	NUCR4	MPER, ERC, HEXASC, TDER, TDDER, TER, BER, TDAER, BG, HDGINP, XYSSET, MAP, DISDEL, NRWCOR, NEWCDO, MPRDY2, ASCFIX, INDEX, LPTOUT.
NU4EVD	NUCR4	NU4EVD is linked to any block using routines in the NUCR4 block.
TDEV	TDXFM	TDXFM, TDTXY1, FPASC, TRNF, XMOVO, XMOVOP.
TDEVD	TDXFM	TDEVD is linked to any block using routines in the TDXFM block.
NCEV	NUCR	ATCTO, ATCTS, TIMTGO, ANLABL, ANVARS PROUTH, PROUT, PCRLF, CATCT, UNPAC, REPAC, TDADD, and DSPDAT.
NCEVD	NUCR	NCEVD is linked to any block using routines in the NUCR block.
NCSEV	NUCR2	STMU, VESSIM, VESSIN, STUP, RDNDEG, TDSP, and SFASC.
NCSEVD	NUCR2	NCSEVD is linked to any block using routines in the NUCR2 block.
HKEV	HKBD	HKBD, BUFC, VESSCL, VESDRW, and VESSYM.
HKEVD	HKBD	HKEVD is linked to any block using routines in the HKBD block.

b. The software math package, which is part of the autoplot feature contained in the HP graphics code, was not required by the PILOT program. To eliminate the software math package, the code from address B800 H thru BFFF H is deleted and the code at addresses 631B H, 631C H, and 631D H is changed to 0 H.

3.2.2 **DATCF Module:** The data communications segment of the HP software is not required by the PILOT software. In the HP 2649A terminal, the entire 64K of memory is used by the software shipped with the terminal. To accommodate PILOT software, unused portions of the HP software were eliminated. DATCF is a software patch which replaces the data communications segment for PILOT software in the HP terminal. HP Data Communication software resides in memory locations 5000 H to 5FFF H. DATCF resides in memory locations 5000 H to 520B H and leaves memory locations 520C H to 5FFF H available for PILOT software, which equals 3572 available bytes.

3.2.3 **RAM Allocation:** The HP 2648 software has 16K bytes of RAM installed. The PILOT program requires only 8K of RAM, 4K for HP code and 4K for PILOT code. To convert the unused 8K of RAM to ROM the following addresses were changed: 156 H to EF H, 15B H to E0 H, and 169 H to E8 H.

3.2.4 **Keyboard Decoding:**

a. The PILOT unit does not use the full keyboard provided by the HP 2649A. A smaller keyboard is used which has only 23 keys. The PILOT keyboard does not require the same keyboard decoding. The PILOT keyboard uses only upper case characters which requires that address 482C H be changed to A7 H. The PILOT keyboard has redefined the HP keys. To accommodate the redefinition of keys the following addresses were changed:

ADDRESS	NEW VALUE	ADDRESS	NEW VALUE
483C H	70 H	4880 H	31 H
483D H	24 H	4881 H	F0 H
4841 H	F0 H	48B8 H	A3 H
4844 H	22 H	4889 H	F1 H
4849 H	F1 H	48BE H	54 H
484C H	5B H	48C0 H	33 H
484D H	26 H	48C1 H	F2 H
484E H	74 H	48C6 H	59 H
4851 H	F2 H	48C8 H	A4 H
4856 H	79 H	48C9 H	F3 H
4859 H	F3 H	48CE H	55 H
485E H	75 H	48D0 H	35 H
4861 H	F4 H	48D1 H	F4 H
4866 H	69 H	48D6 H	49 H
4869 H	F5 H	48D8 H	A2 H
4871 H	F6 H	48D9 H	F5 H
4879 H	F7 H	48E0 H	37 H
4880 H	2E H	48E1 H	F6 H

ADDRESS	NEW VALUE	ADDRESS	NEW VALUE
4885 H	40 H	48E8 H	A1 H
4889 H	8B H	48E9 H	F7 H
488D H	28 H	48F0 H	2E H
488E H	72 H	48F8 H	39 H
4891 H	88 H	48F9 H	8B H
4896 H	65 H	48FE H	52 H
4899 H	8C H	4901 H	88 H
489E H	77 H	4906 H	45 H
48A6 H	71 H	4908 H	2D H
		4909 H	8C H
		490E H	57 H
		4916 H	51 H

b. The HP keyboard is serviced by a 10-millisecond interrupt to determine if the operator has hit any keys. The 10-millisecond interrupt is a precise frequency and, therefore, is an excellent source for driving a clock routine. The HP code is changed to call the PILOT clock routine TIMER. The following addresses were changed: 8F2 H to 33 H and 8F3 H to D8 H.

3.2.5 **Wait-Loop Control:** In normal operation the HP 2648 code initializes the terminal and then enters a wait-loop. From the wait-loop the code can respond to any operator commands or hardware interrupts. The HP code has been changed so that when the wait-loop is entered, control passes to the PILOT code by jumping from the wait-loop to the PILOT code. The following addresses were changed to accommodate this feature: 2A6 H to 30 H and 2A7 H to 50 H.

3.3 PILOT Variables and Constants:

3.3.1 **HCOM Module:** No calls are made to HCOM, only references through external statements. HCOM contains all variables used in the PILOT software.

3.3.2 **HCOMF Module:** Certain constants are used in various locations by the PILOT software, and rather than define a constant every time it is needed, HCOMF defines all constants. Whenever a constant is needed, it is referenced by the name given in HCOMF. No calls are made to HCOMF.

3.4 Keyboard:

3.4.1 PILOT Keyboard Code:

a. The PILOT keyboard uses HP code and code written for the PILOT keyboard. The HP code performs the hardware task of determining when the operator hits a key. The code written for the keyboard enables the operator to command the PILOT to turn ON or OFF the various PILOT options.

b. Figure 3-2 summarizes the operation of the keyboard using a state diagram. Using the keyboard, the operator can enable a particular function, including specifying any quantities needed, or turn OFF a particular function. Should a mistake be made, the entire entry can be cleared.

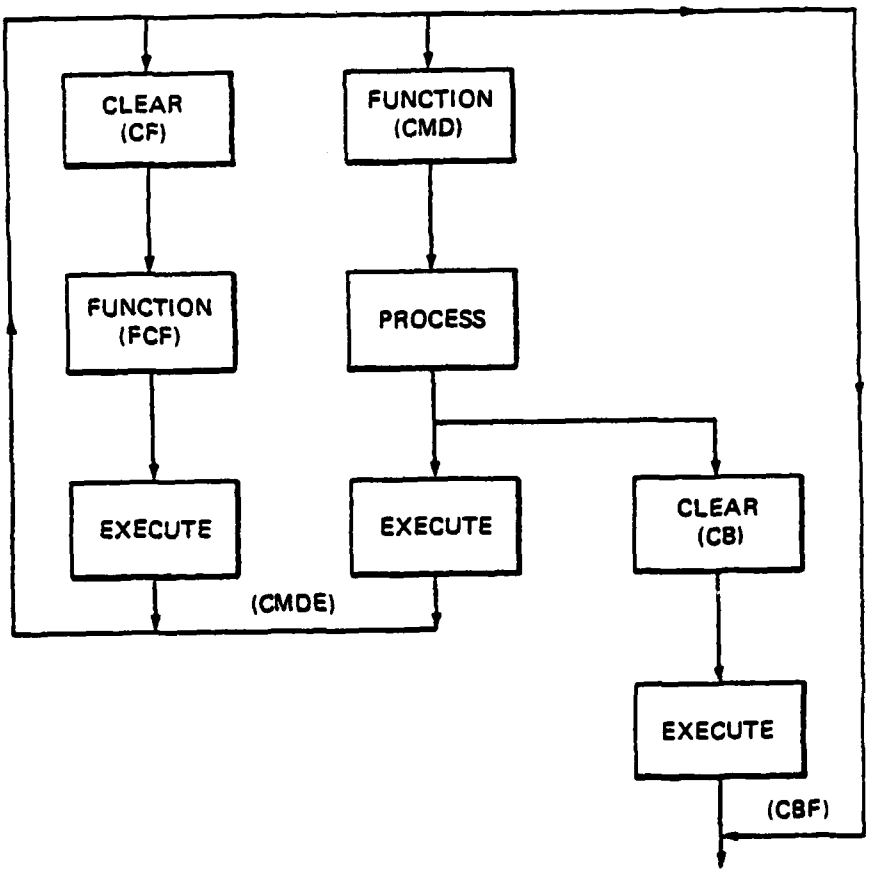


FIGURE 3-2. PILOT KEYBOARD STATE DIAGRAM.

3.4.2 HKBD - Main PILOT Keyboard Decoding Module: The keyboard code uses HKBD to do the main decoding. HKBD uses the flags as defined in Table 3-2 to determine the state of the keyboard and then takes the correct action. Whenever a key is hit, as determined by HP code, HKBD is called (see Fig. 3-3).

TABLE 3-2. KEYBOARD FLAGS.

FLAG	DEFINITION
CMD	Indicates which function to perform.
CMDE	When equal to CMD indicates the function is to be executed. When equal to FCF indicates the function is to be cleared.
CF	Indicates the operator is in the process of clearing a function.
FCF	Indicates which function to clear.
CB	Indicates the command buffer is to be cleared.
CBF	When equal to CB indicates the command buffer is to be cleared.
SHF	Keyboard shift flag.
STAF	Indicates that the STATUS menu is displayed and that the status prompt is displayed.
PJF	Indicates which project is in effect.
MSTF	Indicates if the Master or Detail is displayed.
PF	Indicates if number keys are allowed.

3.4.3 DSF - Display of Function Name Selected by Operator: If a function name is to be displayed, DSF is called. DSF decodes the keyboard entry and displays the corresponding function name in the keyboard display buffer. HKBD calls DSF whenever the function name is to be displayed (see Fig. 3-4).

3.4.4 PRNU - Numeric Key Processor: PRNU handles the processing of numeric keys. HKBD calls PRNU whenever a numeric key is a valid entry. PRNU displays the character in the keyboard display buffer and saves the key in a buffer for execution of the command. Figure 3-5 is a flowchart of this routine.

3.4.5 DISFCF - Clear Display: HKBD calls DISFCF whenever clear is displayed and the operator has selected the function to be cleared. DISFCF displays the function name to be cleared in the keyboard display buffer.

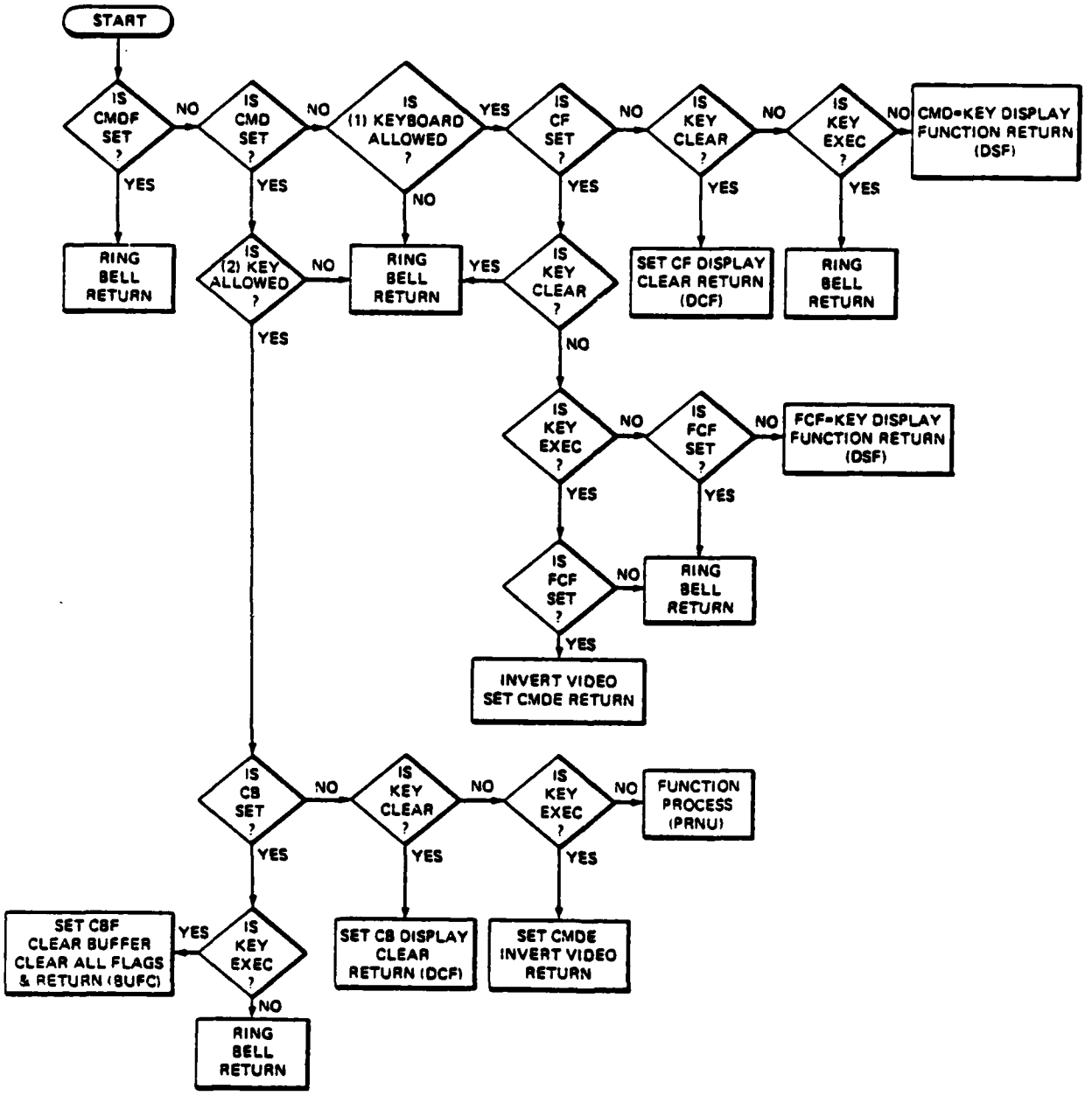


FIGURE 3-3. PILOT KEYBOARD CONTROLLER/DECODER.

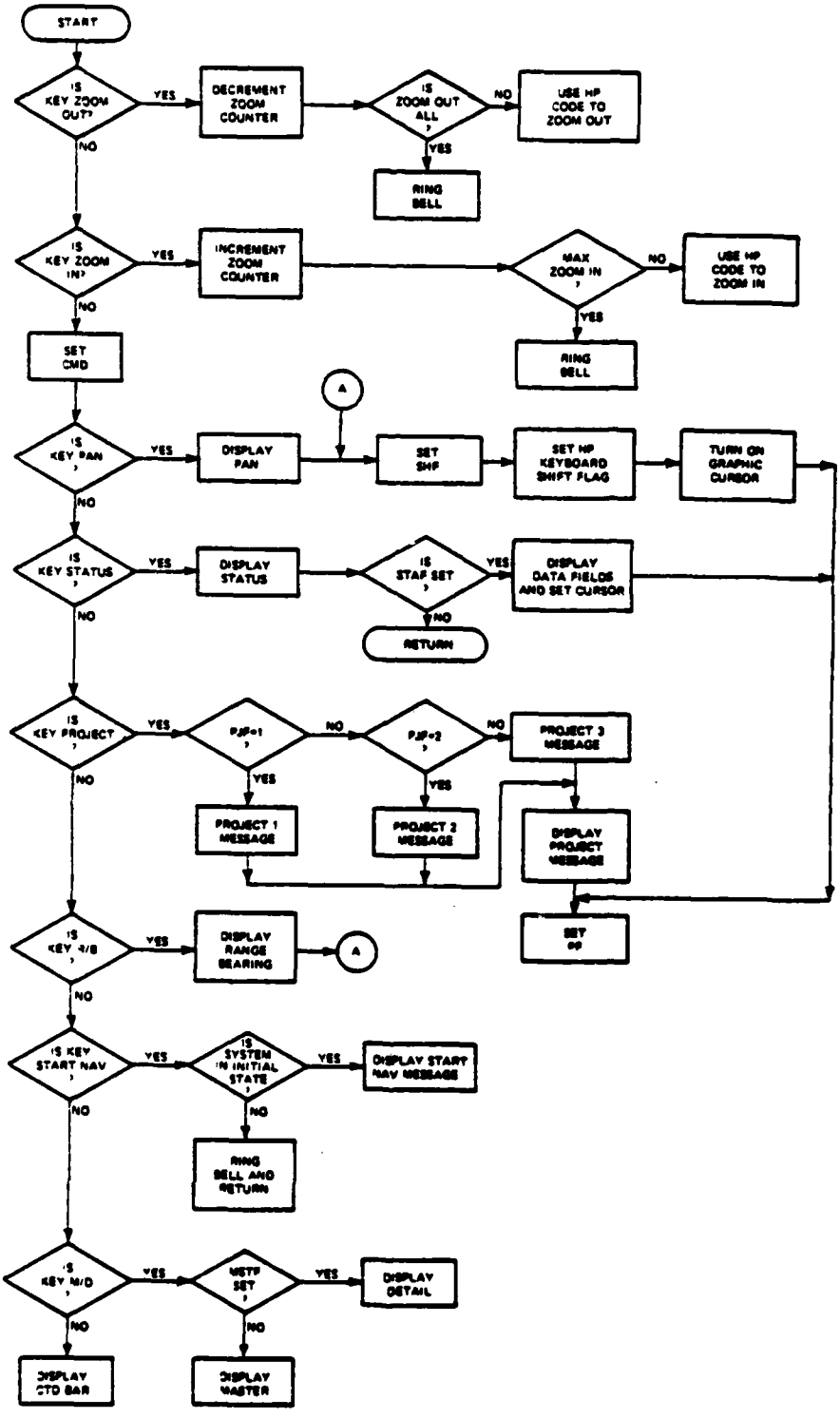


FIGURE 3-4. KEYBOARD DISPLAY FUNCTION SEGMENT.

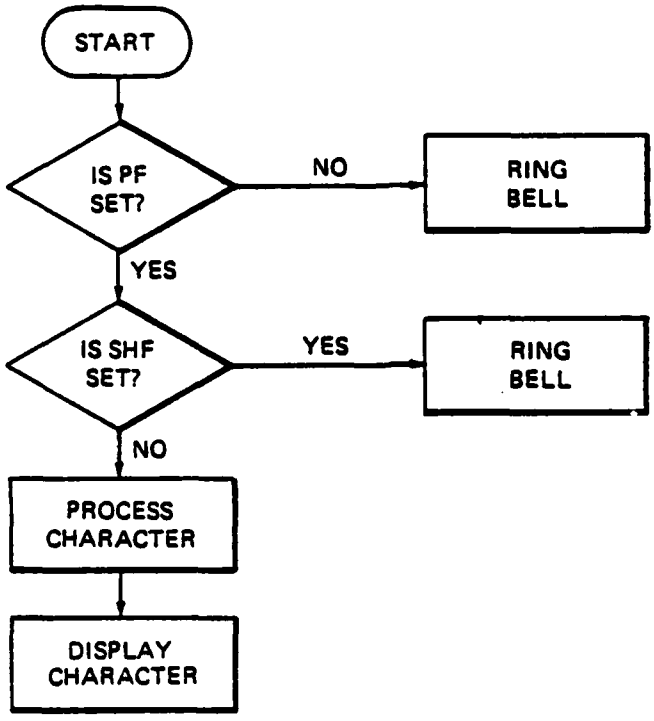


FIGURE 3-5. ROUTINE PRNU.

3.4.6 DCF - Clear Display in Keyboard Display Buffer: DCF displays clear in the keyboard display buffer whenever called by HKBD. DCF checks CMD to determine if clear is to be displayed in the beginning of the buffer or after a function in the buffer.

3.4.7 BUFC - Clear Keyboard Display Buffer: BUFC is called by HKBD whenever the keyboard display buffer is to be cleared, and also resets all keyboard flags.

3.4.8 HKDEC - Keyboard Decoder: The PILOT keyboard code sets various flags to determine what action the PILOT software is to take. HKDEC decodes the keyboard flags and sets the zero and carry flags to determine what action, if any, is to be taken. The C register contains the keyboard value of a function on entry to HKDEC.

3.5 Routine TIMER:

3.5.1 Real-Time Interrupt:

a. Routine TIMER is an interrupt driven real-time clock routine, which assumes a 100 Hz rate and provides time of day (in hundredths of seconds, seconds, minutes, hours) and day of year. It requires an external six-byte memory array (named HSEC) for storage.

b. All data are stored in binary form with day of year requiring two bytes. Day of year is therefore a 16-bit word, where only the LS bit of the MS byte is actually used. The count will go from $(365\text{-}23\text{-}59\text{-}59\text{-}99)_{10}$ to $(01\text{-}00\text{-}00\text{-}00\text{-}00)_{10}$ with no provision for leap years.

c. Initialization of the six-byte memory array (current time) is the responsibility of the operator. Executive time is variable; i.e., the subroutine will return to program control after the last sequential parameter (1/100's of a second, second, etc.) update.

3.5.2 GCP - Graphic Cursor Update: The graphic cursor position (GCP) is used as a reference point for locating the vessel on the display. GCP updates the position of the graphic cursor using HP software. GCP is called by the executive to maintain the position of the graphic cursor.

3.6 Executive:

3.6.1 PILOT Executive:

a. The PILOT software consists of two basic parts: initialization and navigation loop. The PILOT unit remains in the initialization mode until the operator instructs the unit to "start navigation." At that time the unit is placed in the navigation loop permanently or until the operator restarts the unit.

b. The PILOT executive, EXEC, is the program which is jumped to from the HP wait-loop. The PILOT executive is the main controller and decision maker of the total PILOT software package. A functional block diagram of the executive is shown in Fig. 3-6. Figure 3-6 is to be used in connection with the detailed flow charts shown in Figs. 3-7 through 3-20.

c. The detailed flow charts accurately describe the PILOT executive. The conventions used in the flow charts are: the names within brackets[ ] refer to variables and names within parentheses ( ) refer to subroutine names.

3.6.2 **MDDIS - Display of Background Graphics**: Whenever the background graphics on the PILOT display is updated, MDDIS performs the update function. MDDIS examines the master-detail flag to determine which set of graphics to display. The desired background graphics is then displayed, and the variables that located this display in the real world are also updated. The scale, rotation angle, and X and Y coordination of the display are updated to reflect the values associated with the display. MDDIS is used to display the background graphics on initial turn-on, whenever the operator toggles from the master chartlet to the detailed chartlet, or from the detailed chartlet to the master chartlet. MDDIS also turns off such features as zoom and range/bearing.

3.6.3 **Tape Handlers**:

3.6.3.1 Updating Master-Detail File Information:

3.6.3.1.1 Routine MDU: MDU updates the master and detailed chartlet information in memory by reading a master and detail file from tape. MDU uses MDRT to perform the actual reading from tape of the master and detail files. MDU assumes that PILOT has reached the previous trip point and handles all the necessary updating after reading the files from tape. DU is an entry point into MDU for only updating the detailed chartlet information.

3.6.3.1.2 Routine MDRT: MDRT locates the file number given in the accumulator and reads the file into the address contained in the HL register pair. MDRT uses RAF to perform the actual tape reading function.

3.6.4 **HPI - Hewlett-Packard 2649A Initialization**: The standard HP code performs the initialization of the HP 2649A terminal. The terminal is left in a ready state from which the operator can perform any task. The PILOT software takes control of the code at this point. HPI is called to clear the CRT and turn off the alphanumeric cursor.

3.6.5 **Status**:

a. When the status display is on, changes can be made to the status menu by hitting the STATUS key, entering the appropriate data, and then hitting the EXECUTE key. After this sequence has occurred, the PILOT executive calls subroutine STUP to decode the status menu input buffer and update the selected status variable.

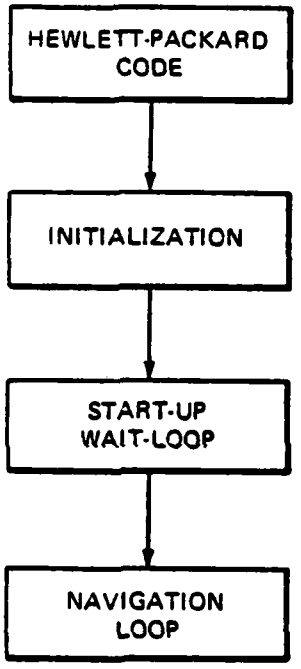


FIGURE 3-6. PILOT EXECUTIVE.

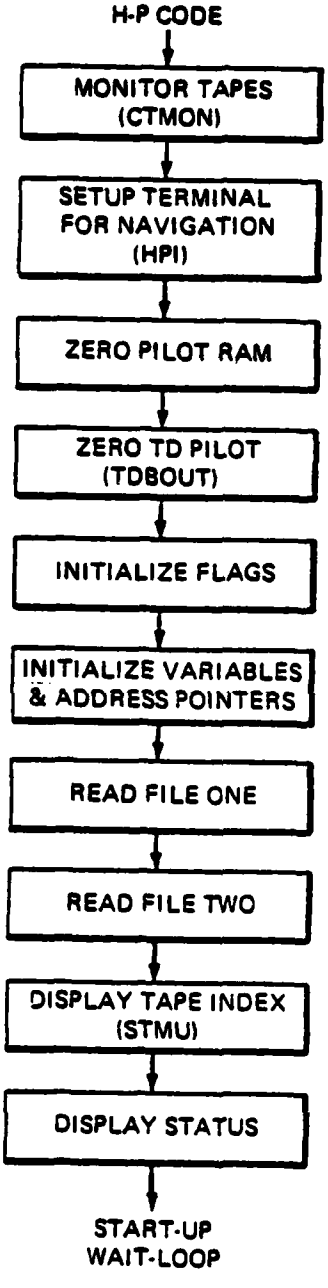


FIGURE 3-7. INITIALIZATION.

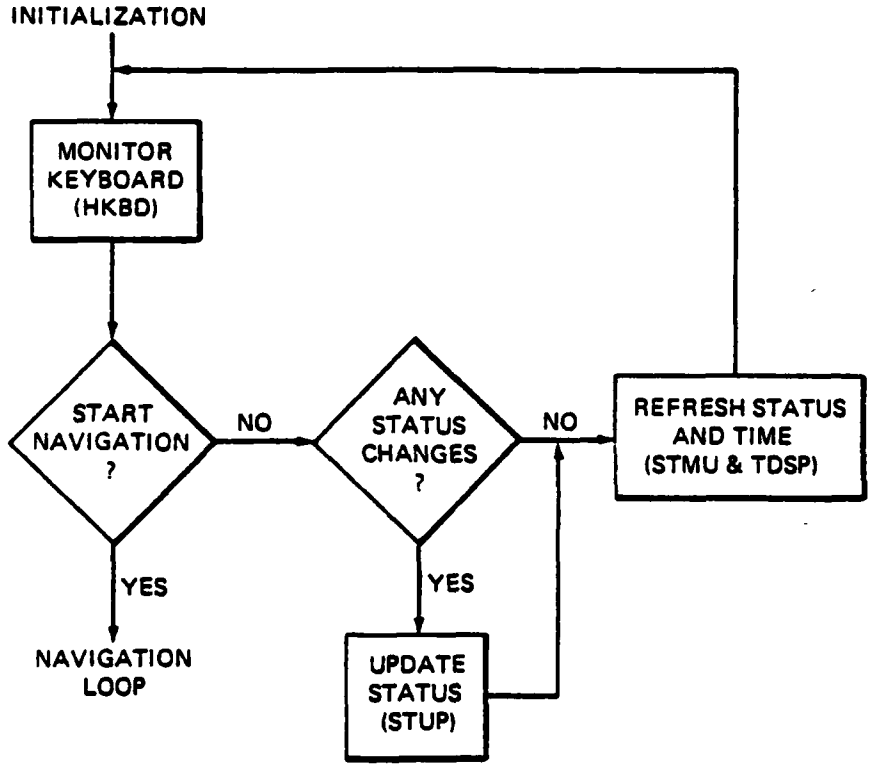


FIGURE 3-8. START-UP WAIT-LOOP.

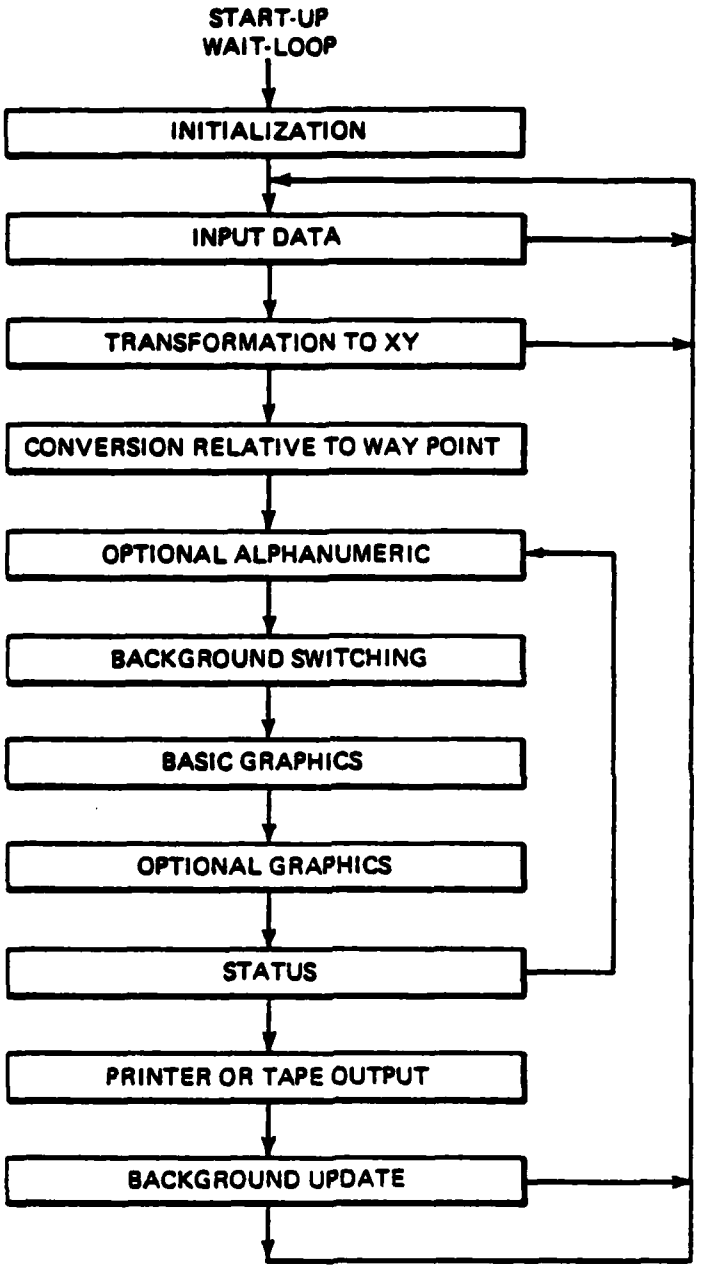


FIGURE 3-9. NAVIGATION LOOP.

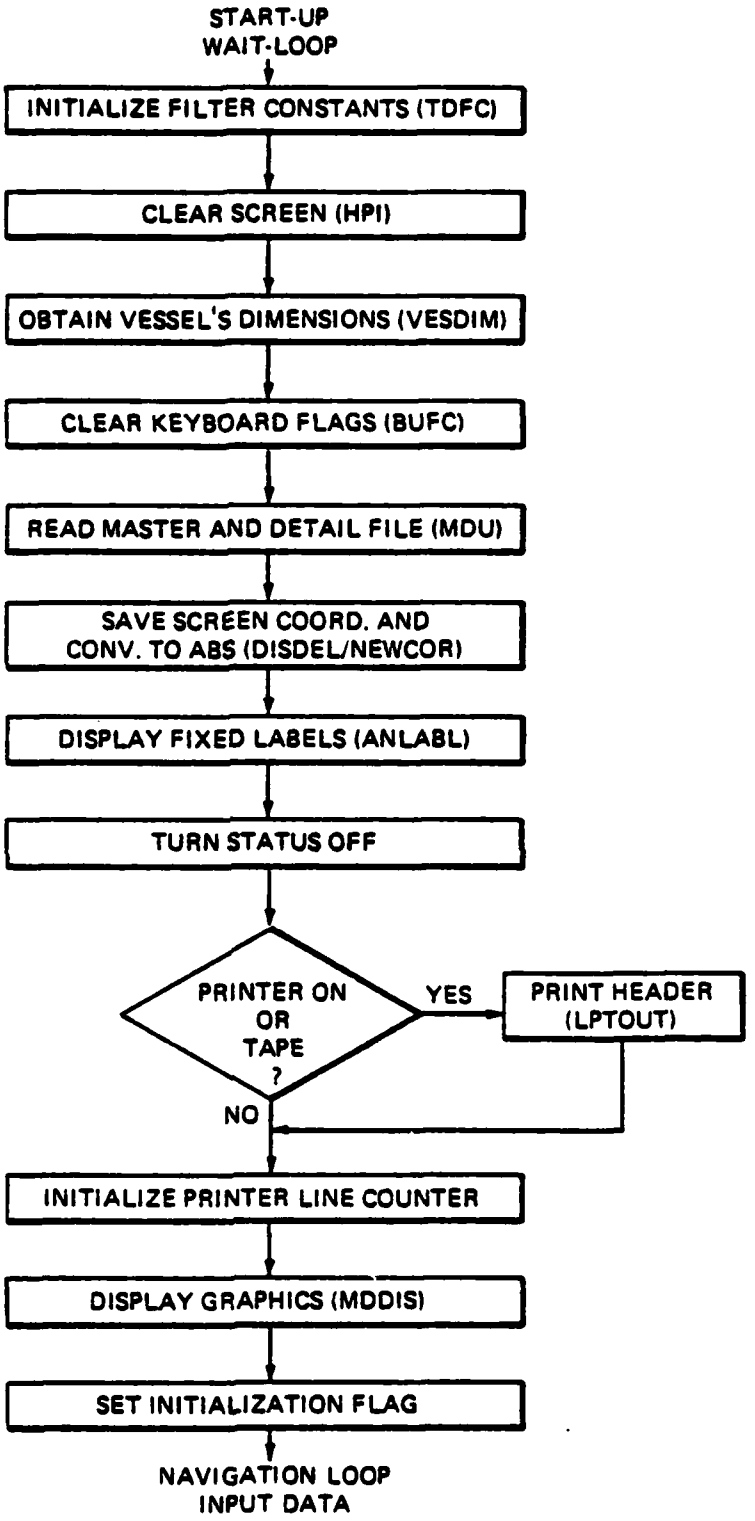


FIGURE 3-10. NAVIGATION LOOP INITIALIZATION.

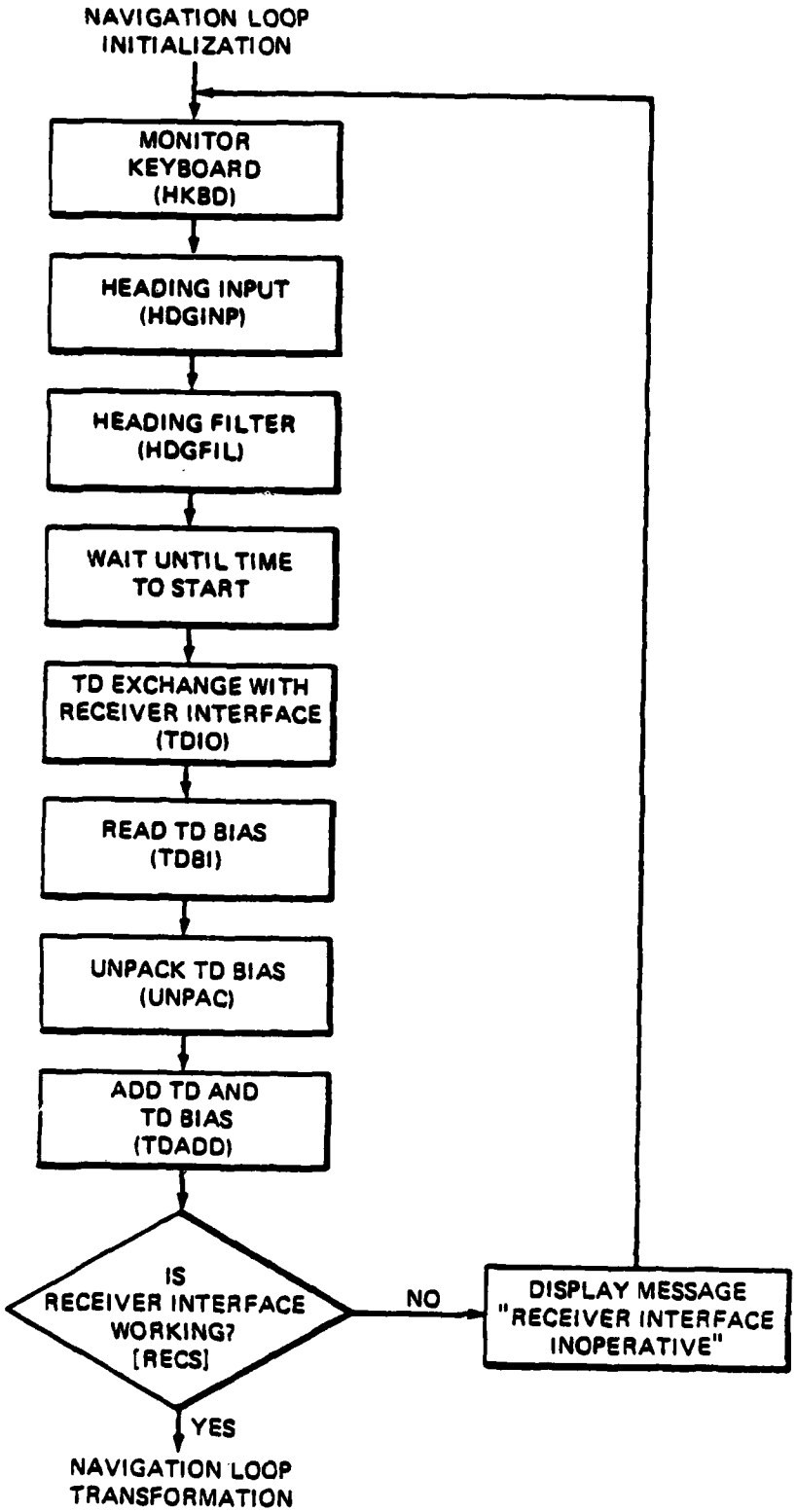


FIGURE 3-11. NAVIGATION LOOP INPUT DATA.

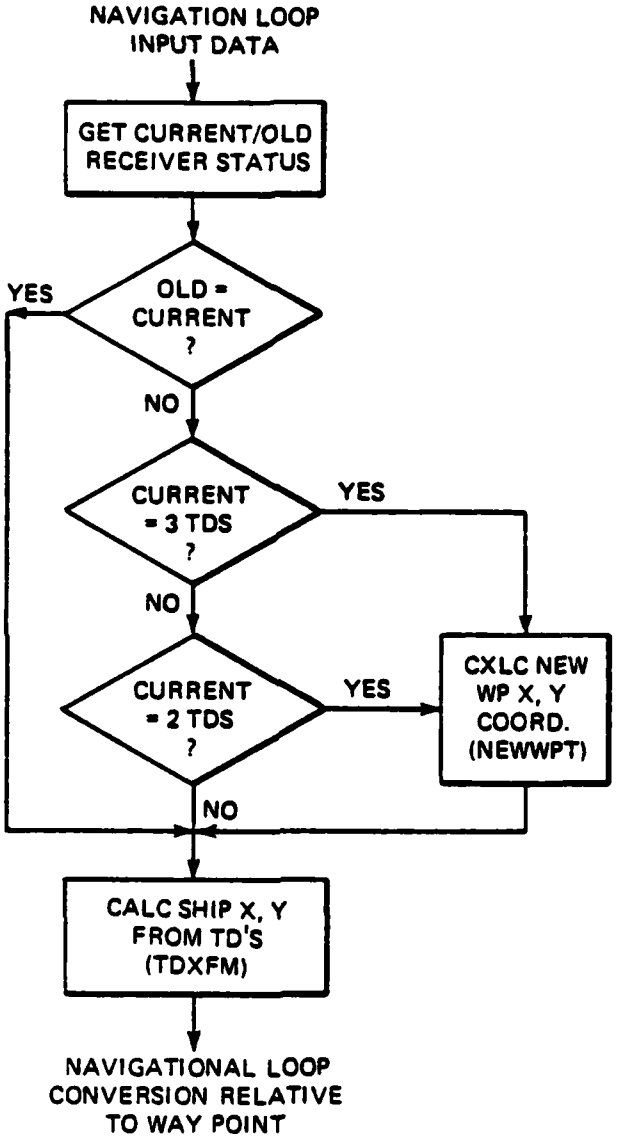


FIGURE 3-12. NAVIGATION LOOP TRANSFORMATION TO XY.

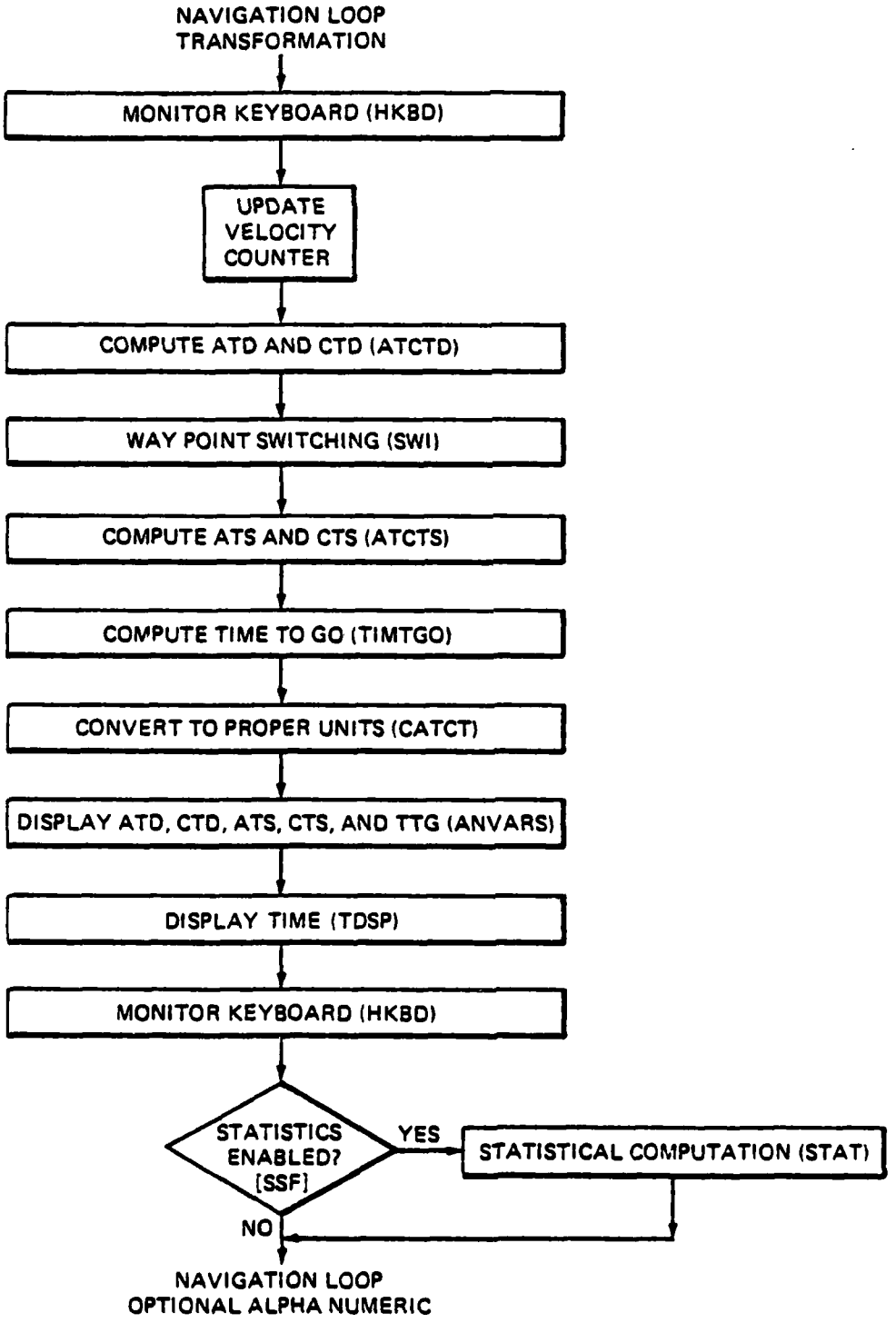


FIGURE 3-13. NAVIGATION LOOP RELATIVE TO WAYPOINT.

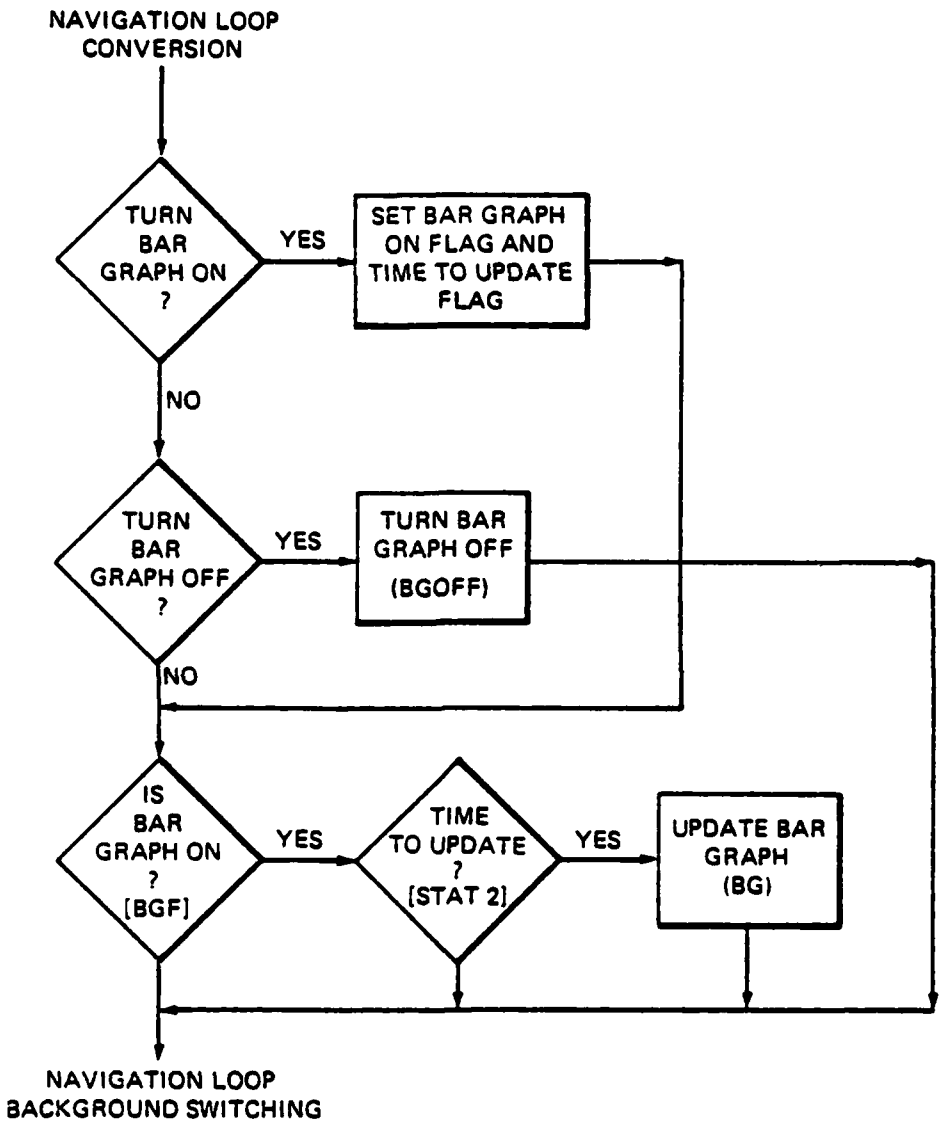


FIGURE 3-14. NAVIGATION LOOP OPTIONAL ALPHANUMERIC.

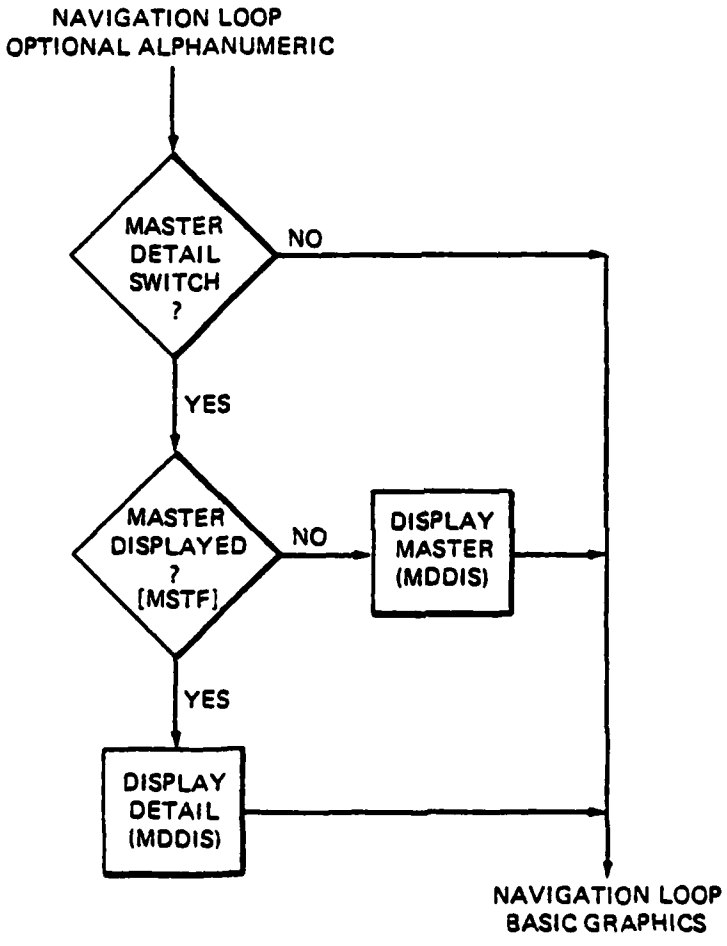


FIGURE 3-15. NAVIGATION LOOP BACKGROUND SWITCHING.

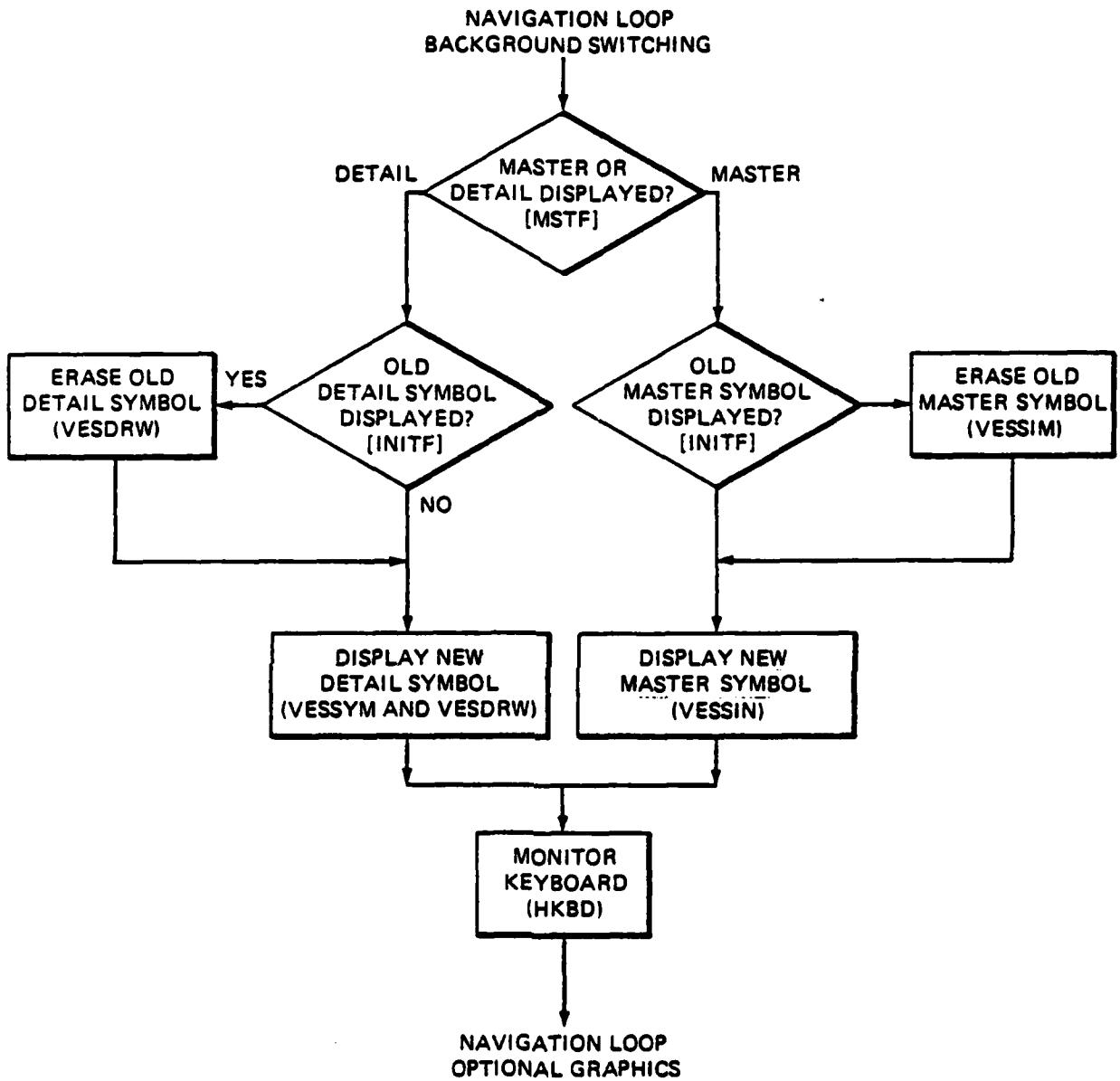


FIGURE 3-16. NAVIGATION LOOP BASIC GRAPHICS.

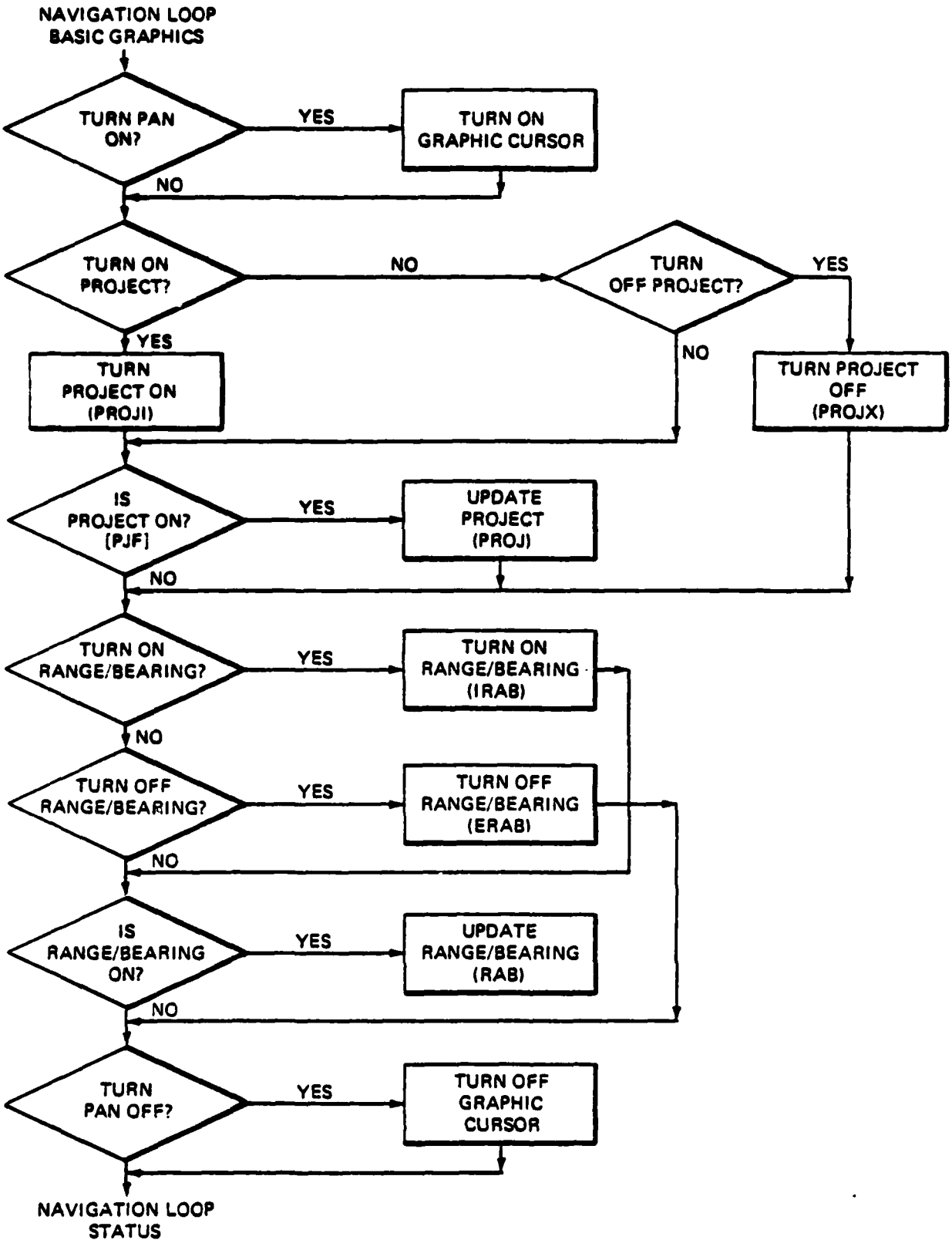


FIGURE 3-17. NAVIGATION LOOP OPTIONAL GRAPHICS.

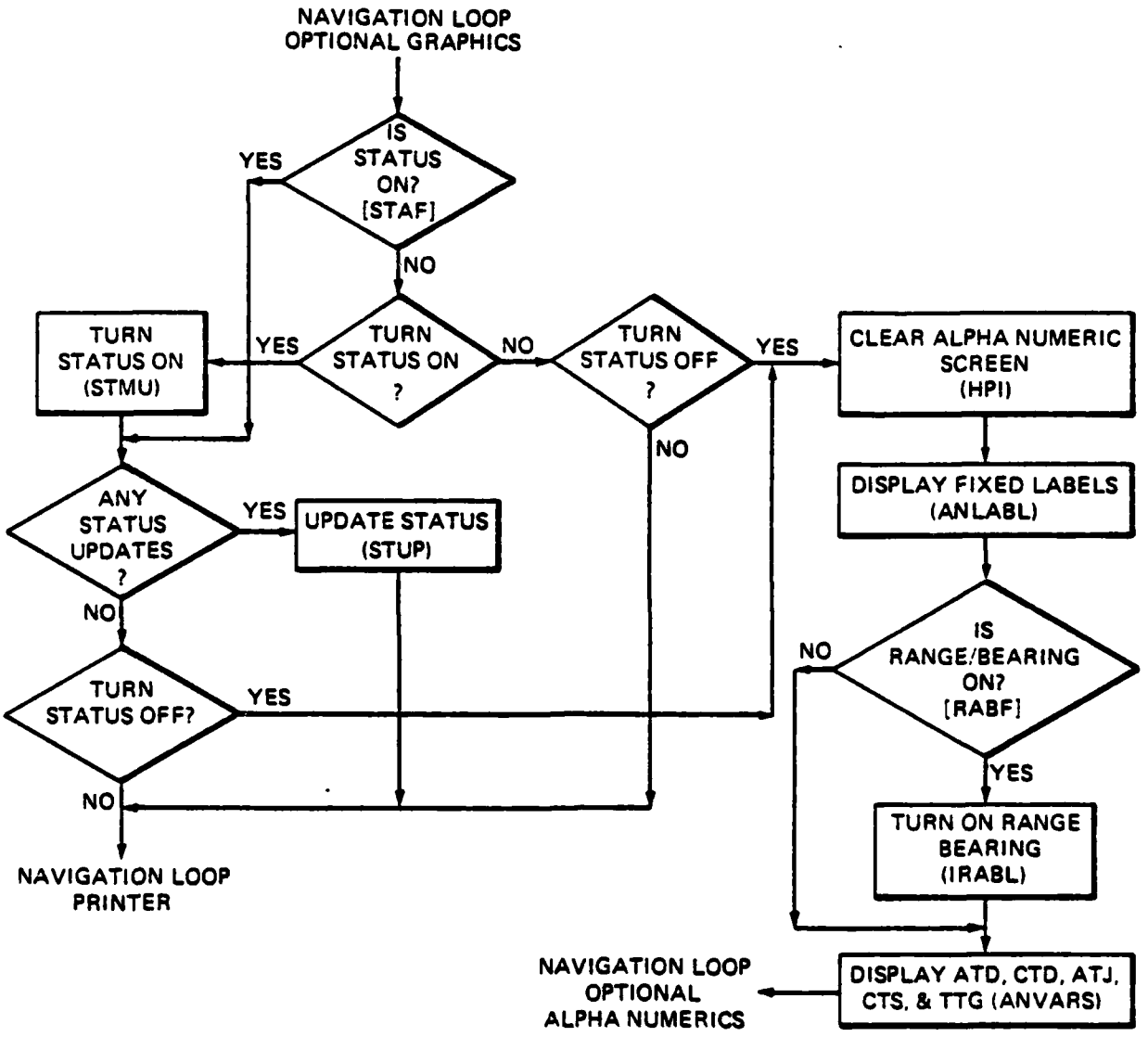


FIGURE 3-18. NAVIGATION LOOP STATUS.

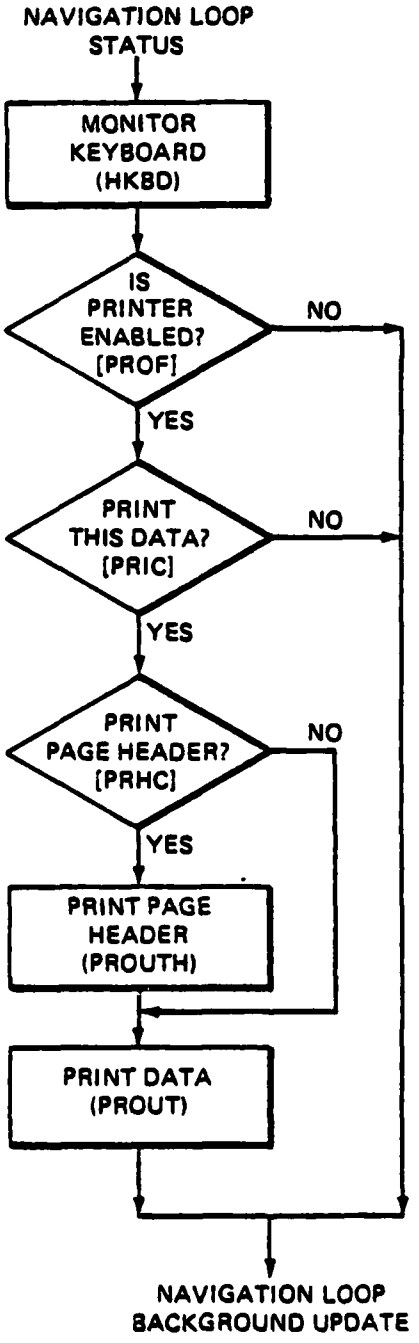


FIGURE 3-19. NAVIGATION LOOP PRINTER OR TAPE OUTPUT.

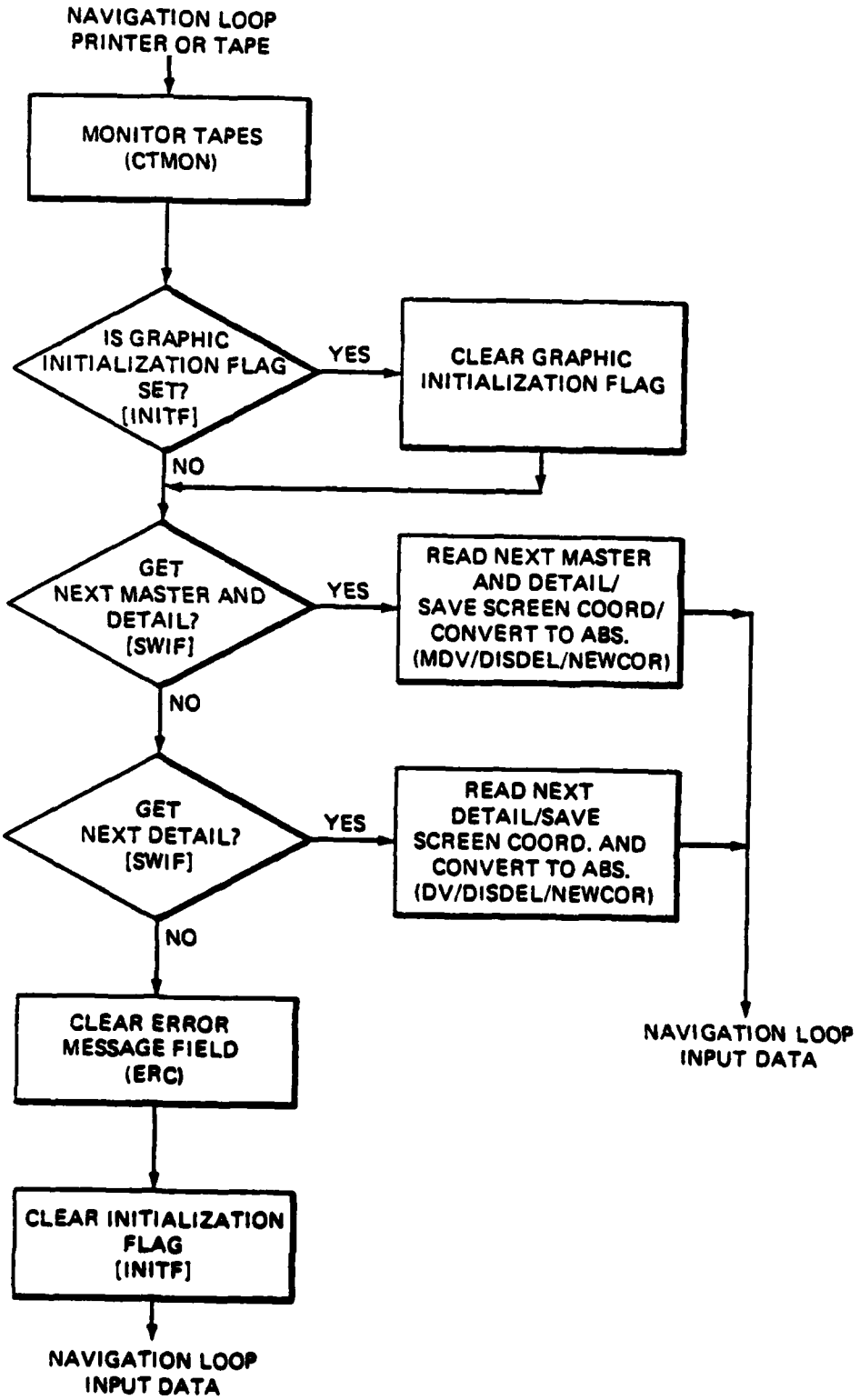


FIGURE 3-20. NAVIGATION LOOP BACKGROUND UPDATE.

b. The status input data is stored in buffer KBUFA. The data appears in the following format:

P P S D D ... D D E

where P is the two-digit parameter number, S is a space character, D are the data characters, and E is an end of page character.

c. When STUP is called, buffer KBUFA already contains the status input data in the above format. STUP will first decode the parameter number, and then process the entered data. If an illegal parameter number is entered, or if an illegal data string is entered for the given parameter, then STUP will return without changing the status menu. When a correct entry is made, STUP will change the given status variable and take any other appropriate action.

d. Figure 3-21 shows the nine status menu parameters that can be changed by the operator. Time data, parameter 10, must be entered as hours (one or two digits), then a decimal point, followed by minutes (one or two digits). If only hours is entered, and no decimal point or minutes following, then the minutes will be set to zero. The way-point file number, parameter 20, is entered as a one to three digit number. The three TD bias numbers, parameters 31, 32, and 33, are entered as a sign character (no sign for a positive number) followed by a one- to four-digit number. The TD bias data is also output to the hardware after being decoded.

e. The four flags, project (parameter 40), filter (parameter 50), print (parameter 60), and test (parameter 70) are entered as a one-digit number which cannot be greater than three. STUP will change the filter constants if the filter flag is changed. Changing the test number will cause the appropriate test procedure to be executed.

3.6.5.1 STMU - Status Menu Display Drive:

a. The status menu is output to the terminal display by calling subroutine STMU. STMU is the driver subroutine which calls the six subroutines that format and display the data for the three status menu groups.

b. STMU calls subroutines STMUAD, STMUBD, and STMUCD which convert the status menu data for groups A, B, and C, respectively, into ASCII and store it in the appropriate variables. The remaining subroutines described below perform various ASCII conversion functions, and are called by STMUAD, STMUBD, and STMUCD.

c. STMU also calls subroutines STMUA, STMUB, and STMUC which output the status menu data for groups A, B, and C to the terminal display. These three subroutines are described in Sections 3.6.5.2 through 3.6.5.4 and Fig. 3-22.

d. STMU does not output the status menu time of day data. This is performed by calling subroutine TDSP, which formats and displays the time of day data in the upper right-hand corner of the screen. STMU does display

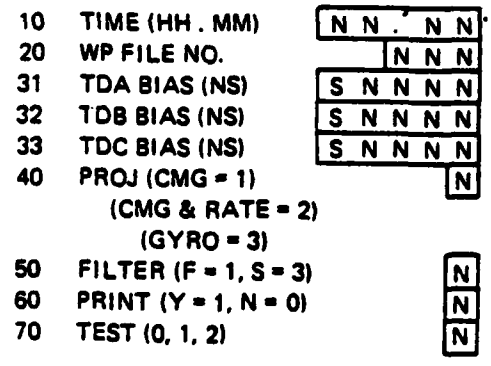


FIGURE 3-21. STATUS MENU.

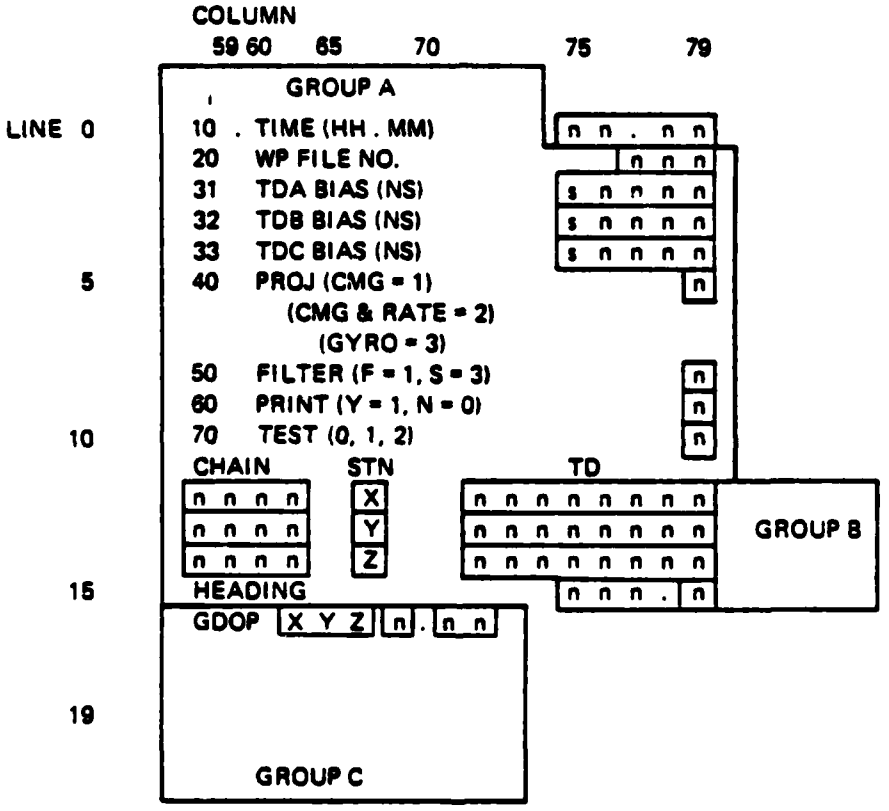


FIGURE 3-22. STATUS MENU - GROUPS A, B, AND C.

the fixed TIME label every time it is called. The PILOT executive calls STMU and TDSP to display the status menu and the time of day data. The fixed data in the status menu is only output when the initialization flag (INITF) is not zero.

3.6.5.2 STMUAD - Status Menu Group A Data: STMUAD converts status menu group A data into ASCII. The ASCII data are stored in data tables STV1 and TMS1. STMUAD is called by STMU.

3.6.5.3 STMUBD - Status Menu Group B Data: STMUBD converts status menu group B data into ASCII. The ASCII data are stored in data table TD1. The heading angle is rounded off to the nearest 0.1 degree. STMUBD is called by STMU.

3.6.5.4 STMUCD - Status Menu Group C Data: STMUCD converts status menu group C data into ASCII. The ASCII data are stored in data table TMS1. STMUCD is called by STMU.

3.6.5.5 SFASC - Single Fixed to ASCII Conversion: SFASC converts single fixed-point binary data (2 bytes) into ASCII. The number of characters desired in the ASCII buffer is an entry variable. Leading spaces are added to the ASCII buffer. The single fixed-point number is assumed to be positive.

3.6.5.6 B2ASC/B3ASC - Binary Fixed to ASCII Conversion: B2ASC/B3ASC converts a one-byte binary number into ASCII. B2ASC generates two ASCII characters, while B3ASC generates three ASCII characters. Leading spaces are added to the ASCII buffer.

3.6.5.7 SGDOP - Format GDOP Data for Status Display: SGDOP converts one floating-point GDOP number into the correct ASCII format for the status display. The GDOP format consists of one integer character, an end-of-page character, two fractional characters, and an end-of-page character. SGDOP is called by STMUCD.

3.6.5.8 D1ASC - Convert 1 BCD Character to ASCII: D1ASC converts one BCD character into ASCII. No check is made for illegal BCD characters. D1ASC is called by STMUAD.

3.6.5.9 D4ASC - Convert 4 BCD Characters to ASCII: D4ASC converts four BCD characters, plus a sign (5 bytes) into ASCII (5 characters). The BCD data consist of a sign byte (0=plus, 1=minus), followed by four BCD characters with the most significant character first. Each BCD character is contained in one byte. If the BCD data are positive, a space is stored in the ASCII buffer sign location. Leading zeros are blanked in the ASCII buffer. D4ASC is called by STMUAD.

3.6.5.10 STEOP - Store End-of-Page Character: STEOP stores an end-of-page character at the location indicated by register pair DE. The end-of-page character = CEH. Increment register pair DE.

3.6.5.11 STSPC - Store Space Character: STSPC stores a space character (20 H) at the location indicated by register pair DE. Increment register pair DE.

3.6.6 TDSP - Display Time-of-Day on Screen: TDSP converts the time-of-day data to ASCII and displays it on the terminal screen. TDSP displays the time as HRS. MIN in the upper right-hand corner of the screen. TDSP is called by the executive.

3.6.6.1 STMUA, STMUB, and STMUC - Status Menu Groups A, B, and C. Routines STMUA, STMUB, and STMUC are modules that paint the status menu on the CRT in the right-hand portion of the screen. The variables must be in ASCII format with certain required constant characters. STMUA, STMUB, and STMUC paint groups A, B, and C portions of the status menu (see Fig. 3-23).

3.6.7 Input:

3.6.7.1 Receiver Interface Communications in the Intel 8080: There are four basic routines (TDD, TDIDEN, TDFC, and TDIO) which handle the communications between the Intel 8080 and the receiver interface. These routines are described in the following sections.

3.6.7.1.1 TDD - Diagnostic Command to Receiver Interface: Routine TDD sends a diagnostic command, the contents of the accumulator, to the receiver interface, which places it in one of the diagnostic modes.

3.6.7.1.2 TDIDEN - Identifying TD's to Receiver Interface: Routine TDIDEN sends the identifying TD's to the receiver interface after sending the correct command of 10 H. TDIDEN uses the TD's which were read from the tape.

3.6.7.1.3 TDFC - Current Value of Filter Constants to Receiver Interface: The current value of the filter constants is sent to the receiver interface by routine TDFC. TDFC examines FILF to determine which filter constants the operator has selected, and sends the command of 18 H to the receiver interface to signify that new filter constants are being sent and that the TD filters should be reset.

3.6.7.1.4 TDIO - TD Exchange With Receiver Interface: Routine TDIO performs the TD exchange with the receiver interface, and sends a command byte of 01 H to signal the receiver interface that a TD exchange is to occur. The receiver interface then sends its status, TD's, and $\dot{TD}$'s to the Intel 8080. The Intel 8080 then sends the $\dot{TD}$'s to the receiver interface; this exchange constitutes one data exchange.

3.6.7.2 HDGINP - Heading Input Module:

a. HDGINP inputs, as memory-mapped I/O, two data bytes from the TD BIAS and HEADING interface card and converts the data to a format compatible with other PILOT software. Heading information can be supplied from one of two sources: the ship's gyro or the Digicourse magnetic compass unit. The TD BIAS and HEADING interface card formats the raw data prior to placing it into absolute address locations 8600 H (MSB) and 8601 H (LSB). The interface card is designed to accept an input from one or the other source and

STV1	
SEQ	FUNCTION
1	X 10 HRS
2	X 1 HRS
3	EOP
4	K 10 MIN
5	X 1 MIN
6	EOP
7	WP X 100
8	WP X 10
9	WP X 1
10	EOP
11	TDA ALGB SIGN
12	TDA X 12
13	X 100
14	X 10
15	X 1
16	EOP
17	TDB ALGR SIGN
18	TDB X 1K
19	X 100
20	X 10
21	X 1
22	EOP
23	TDC ALEB SIGN
24	TDC X 1K
25	X 100
26	X 10
27	X 1
28	EOP
29	PROJ
30	EOP
31	FILTER
32	EOP
33	PRINT
34	EOP
35	TEST
36	EOP

TMS1	
SEQ	FUNCTION
1	CHAIN #1 X 1K
2	X 100
3	X 10
4	X 1
5	EOP
6	STN #1
7	EOP
8	CHAIN #2 X 1K
9	X 100
10	X 10
11	X 1
12	EOP
13	STN #2
14	EOP
15	CHAIN #3 X 1K
16	X 100
17	X 10
18	X 1
19	EOP
20	STN #3
21	EOP
22	3DOP #1 X 100K
23	X 10K
24	X 1K
25	EOP

TD1	
SEQ	FUNCTION
1	TD1 X 10M
2	X 1M
3	X 100K
4	X 10K
5	X 1K
6	X 100
7	X 10
8	X 1
9	EOP
10	TD2 X 10M
11	X 1M
12	X 100K
13	X 10K
14	X 1K
15	X 100
16	X 10
17	X 1
18	EOP
19	TD3 X 10M
20	X 1M
21	X 100K
22	X 10K
23	X 1K
24	X 100
25	X 10
26	X 1
27	EOP
28	GYRO X 1K
29	X 100
30	X 10
31	OZEH
32	EOP
33	GYRO X 1
34	EOP

FIGURE 3-23. STATUS MENU VARIABLES.

not both simultaneously. The interface card appends a string of 1's or 0's in the least significant six or seven bits of the 16-bit data word. The appended 1's or 0's are used by HDGINP to identify the source of the data.

b. The synchro-to-digital data format (gyro input) consists of a 10-bit straight binary data field and a 6-bit source field of appended binary 1's; the data have a 6-bit binary scaling. Likewise, the Digicourse format consists of a 9-bit straight binary data field and a 7-bit source field of appended binary 0's; the Digicourse data have a 7-bit binary scaling.

c. HDGINP performs the following operations: two bytes are read from the interface card using memory-mapped I/O, the source is identified; and depending on the data source, the binary scaling is removed accordingly. The remaining binary number is converted to the math pack floating - point format in navigation degrees, and then a degrees-to-radians conversion is performed. The resulting radian value is stored in memory at locations HDG through HDG43. HDGINP is called directly by the navigation executive.

3.6.7.3 HDGFIL - Heading Filter Module:

a. HDGFIL computes the TD accelerations required by the digital filter in the receiver interface software. The digital filter uses the TD accelerations to compute the LORAN time differences (TD's) and their rates (TD's). HDGFIL is called by the navigation executive program once per navigation cycle.

b. HDGFIL operates as follows. The vessel radial acceleration at the current navigation update is computed as a function of the previous radial acceleration, the present vessel speed, the elapsed time since the last update, and the change in the vessel's heading since the last update. The heading filter equation is given by:

$$A_{Ri} = A_{Ri-1} + c\{V_i/\Delta t_i\,(\psi_i^m - \psi_{i-1}^m) - A_{Ri-1}\} = A_{Ri-1} + cK$$

where:

A_{Ri} = radial acceleration at present time, t_i

A_{Ri-1} = radial acceleration at previous time, t_{i-1}

c = filter constant

V_i = vessel's speed at time, t_i

$\Delta t_i = t_i - t_{i-1}$ = elapsed time since last update

ψ_i^m = measured heading at present time, t_i

ψ_{i-1}^m = measured heading at previous time, t_{i-1}

$K = V_i/\Delta t_i(\psi_i^m - \psi_{i-1}^m) - A_{Ri-1}$

= change in radial acceleration from t_i to t_{i-1}

c. The radial acceleration, a_{Ri}, is used to compute the rectangular acceleration components, $\ddot{X}$ and $\ddot{Y}$, according to the following relationships:

$$a_X = \ddot{X} = a_{Ri} \cdot \cos(\psi_i)$$

$$a_Y = \ddot{Y} = -a_{Ri} \cdot \sin(\psi_i)$$

where:

$a_X = \ddot{X}$ = acceleration in X-direction

$a_Y = \ddot{Y}$ = acceleration in Y-direction

ψ_i = vessel's "course-made-good" angle

a_{Ri} = just-computed radial acceleration.

The quantity, ψ_i, the "course-made-good" is a computed angle, whereas ψ_i^m and ψ_{i-1}^m are measured angles taken from the ship's gyro.

d. The acceleration components, a_X and a_Y, are used to compute the TD accelerations. The TD accelerations are given by:

$$TDA = g_X(TDA) \cdot a_X + g_Y(TDA) \cdot a_Y$$

$$TDB = g_X(TDB) \cdot a_X = g_Y(TDB) \cdot a_Y$$

$$TDC = g_X(TDC) \cdot a_X + g_Y(TDC) \cdot a_Y$$

where:

TDj = TD acceleration of the j^{th} TD, j = A,B,C

$g_X(TDj)$ = gradient of the j^{th} TD in the X-direction, j = A,B,C

$g_Y(TDj)$ = gradient of the j^{th} TD in the Y-direction, j = A,B,C

a_X = acceleration in X-direction

a_Y = acceleration in Y-direction.

e. The program varibles have the following names. AR and ARO are used for A_{Ri} and A_{Ri-1}, respectively. HFK1 is used as the heading filter constant, C. V is the vessel's speed, V_i, and is computed and stored by the ATCTS module. Time is the variable associated with t_i. HDG is the present measured heading, θ_i^m, while HDGO is the previous measured heading, θ_{i-1}^m. The variable DACCEL is used for K, the change in radial acceleration. The acceleration variables are AX and AY for a_x and a_y respectively. CMG is the vessel's course-made-good angle. The TD accelerations are defined by ATDA, ATDB, and ATDC for TDA, TDB and TDC, respectively. The gradients are given by: GXTDA and GYTDA for the TDA gradients; GXTDB and GYTDB for the TDB gradients; and GXTDC and GYTDC for the TDC gradients. The gradient matrix is stored on the cassette tape and a new set of values are read in whenever a new master file is read from the tape. All variables are floating-point variables.

f. The value of K, the change in radial acceleration, is compared to a threshold value to determine if the vessel is turning. The threshold value is defined by the variable TACCEL. If the value of K (=DACCEL) is less than or equal to the threshold value, the TD accelerations are set to zero.

g. On the first pass through HDGFIL, the previous measured heading, HDGO, is set equal to the present measured heading, HDG.

h. The previous radial acceleration, ARO, is set equal to the present radial acceleration, AR. The TD accelerations are set to zero.

i. Figure 3-24 shows how HDGFIL operates.

3.6.8 TD Bias:

3.6.8.1 TD Bias - Variable Transfer Routines (TDBI and TDBOUT): TDBI and TDBOUT enable the 16-byte transfer of TD Bias variables to and from memory. The external data access point is defined as the address at 8680 H.

3.6.8.2 Conversion Routines:

3.6.8.2.1 UNPAC/REPAC - Conversion of 2-Byte Words to 5-Byte Words and Inverse Operations:

a. UNPAC: Routine UNPAC converts three 2-byte words into three 5-byte words. The input format is:

SAAABBBB

CCCCDDDD

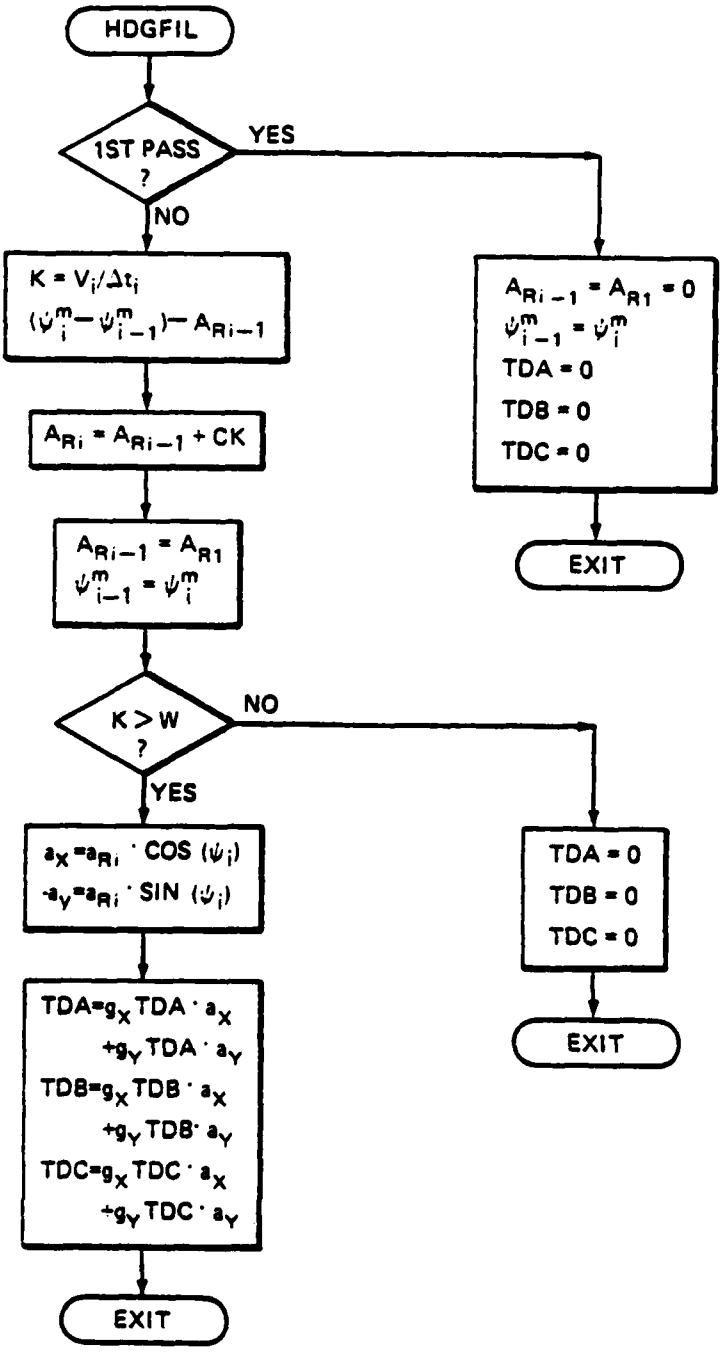


FIGURE 3-24 HDGFIL FLOW CHART.

where S is a sign bit and A, B, C, and D are BCD digits. The output format is:

0000000S

00000AAA

0000BBBB

0000CCCC

0000DDDD

Input data are stored in six sequential memory locations starting at address BIAS. Output data are stored in 15 sequential locations starting at address (BIAS + 6).

b. REPAC: Routine REPAC performs the inverse operation of UNPAC. The input is three 5-byte words stored in 15 sequential memory locations starting at (BIAS + 6). The output is three 2-byte words stored in six sequential memory locations starting at address BIAS.

3.6.8.2.2 TDADD - Convert BCD to Binary and Store:

a. TDADD operates on the three time differences (TD's): TDA, TDB, and TDC. For each TD a 32-bit binary bias is added to the 32-bit TD and the sum is stored in the original 4-byte memory locations. The sequence of operations performed for TDA is as follows: (1) convert 5-byte BCD bias (4 data bytes plus sign) to a 32-bit 2's complement binary number, (2) addition of the 32-bit 2's complement binary number to TDA, and (3) storing the 32-bit sum in four sequential memory locations for TDA. This sequence is repeated for TDB and TDC.

b. The three 5-byte bias data are stored in 15 sequential memory locations starting at (BIAS + 6). The three 4-byte (32 bit) TD values are stored in 12 sequential memory locations starting with TDS.

3.6.9 Transformation:

a. The PILOT software which converts receiver TD values to X-Y Cartesian coordinate values is based on the Loran navigation transformation algorithm taken from Ref. 7. The resulting assembly language programs which are explained in the following sections make use of a new math pack routine called MAP. The calculations and parameters used to perform the transformation are taken from a listing of the FORTRAN subroutine CGQD50, which is also thoroughly documented in Ref. 7.

b. A summary of the PILOT transformation algorithm taken from Ref. 7 follows:

Given:

t = vector containing a set of TD's

$\dot{t}$ = vector containing TD change rate

z_o = vector containing a set of initial coordinate values

h = function to relate t and z

Find:

z = set of coordinates for t

$\dot{z}$ = the rate at which z is changing

Compute:

$$t_o = h(z_o) = \begin{vmatrix} h_1(z_o) \\ h_2(z_o) \\ \cdot \\ \cdot \\ \cdot \\ h_m(z_o) \end{vmatrix}$$

$$L_o = \begin{vmatrix} \frac{\partial h_1}{\partial x} & \frac{\partial h_1}{\partial y} \\ \frac{\partial h_2}{\partial x} & \frac{\partial h_2}{\partial x} \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \frac{\partial h_m}{\partial x} & \frac{\partial h_m}{\partial y} \end{vmatrix} \qquad z = z_o$$

$$L_o^{-1} = (L_o^T L_o)^{-1} L_o^T$$

$$z_1 = z_o + L_o^{-1}(t - t_o)$$

For $K \geq 2$:

$$z_k = 2z_{k-1} - z_{k-2} - L_o^{-1}(h(z_{k-1}) - h(z_{k-2}))$$

$$z = \lim_{k \to \infty} Z_k \qquad \text{(k is actually limited to 32)}$$

Compute L_k and L_k^{-1} at the converged value $z = z_k$. Then

$$\dot{z} = L_k^{-1}\dot{t}$$

3.6.9.1 Transformation Routines:

a. XYSSET: Routine XYSSET stores reference file No. 2 X-Y coordinates for transmitter locations in appropriate RAM locations to allow for station drop out compensation. XYSSET is called by the PILOT executive after reading reference file No. 2.

b. TDXFM: Routine TDXFM is the executive entry point to convert ownship's TD's to X-Y coordinates for navigation, calculation of ship velocity, and generation of GDOP.

c. TDXVAR: Routine TDXVAR sets up working variables for use in converting TD's to X-Y coordinates of position and sets graphics display to indicate receiver status. TDXVAR is called by the TDTXY7 module.

d. TDTXY2: Routine TDTXY2 is the TD to X-Y coordinate conversion submodule used to calculate a new X-Y coordinate position during an iteration. It also determines if minimum convergence criteria are met to terminate the iteration process.

e. TDTXY3: Routine TDTXY3 is the TD to X-Y coordinate conversion submodule that calculates station arcs and TD estimates for the iteration process.

f. TDTXY4: Routine TDTXY4 is the TD to X-Y coordinate conversion submodule used to generate G and A matrix elements.

3.6.9.2 GETARG/PUTARG - General Purpose Utilities:

a. Routine GETARG: GETARG is a general purpose utility used to move data from one source location to destination by using labels which follow a subroutine call. This routine expects the HL register to contain a pointer to source addresses and the DE register to contain a pointer to destination addresses. All data items are expected to be 4-bytes long.

b. Routine PUTARG: PUTARG is the inverse of GETARG and expects the HL register to contain pointer to destination addresses and the DE register to contain pointer to source addresses. All data items are expected to be 4-bytes long.

3.6.9.3 XMOVO - Transform Move Module: XMOVO is a general purpose utility module constituting part of the TD transformation software. It moves four bytes of data from one location to another in memory.

3.6.10 **Along Track and Cross Track AT and CT Computations:**

a. ANLABL - Alphanumeric Labels Module:

(1) Routine ANLABL generates the labels for the display of alphanumeric navigation data. ANLABL and ANVARS are the two modules constituting the alphanumeric display software.

(2) The PILOT system provides an operator with an alphanumeric display of pertinent navigation parameters. These parameters are the vessel's along-track and cross-track distances (ATD and CTD), the along-track and cross-track speeds (ATS and CTS), the time-to-go (TTG), and the waypoint and bearing to which the vessel is headed or from which it is leaving. These data are displayed on the left side of the CRT screen.

(3) The alphanumeric display consists of two parts: the labels and units for the parameters and the actual values of those parameters. The labels are static. The parameter values are variable and are updated during each navigation cycle. ANVARS, the other module, generates the variable data to the display and is called every navigation cycle.

(4) The ANLABL routine is called once during program initialization at start up or reset. It places on the CRT screen the destination waypoint and bearing angles (TO, FROM) and the labels and units for the parameters.

b. ANVARS - Alphanumeric Variables Module:

(1) Routine ANVARS generates the variables for the display of alphanumeric navigation data and is called every navigation cycle.

(2) Before displaying the variable data, ANVARS converts the data from its floating-point format into ASCII string format suitable for display on the CRT. ANVARS writes the ASCII string representations of the parameters into the alphanumeric memory.

c. ANDSP - Alphanumeric Display Message Module:

(1) ANDSP is a utility module that writes an ASCII string into the alphanumeric display memory for display on the CRT screen. It is used to place alphanumeric data on any location on the CRT.

(2) The inputs required for ANDSP are the starting address of the ASCII string and the row and column of where on the CRT the string will start. ANDSP uses these input parameters to setup for the HP-defined DSPMSG subroutine.

(3) The DSPMSG subroutine performs the actual string manipulations and data display operations. It writes the ASCII data into the proper locations of the alphanumeric display memory. Further, DSPMSG is setup by ANDSP to add the message to the normal display rather than erasing the display. DSPMSG assumes that the ASCII string is terminated with an end-of-page character (= 0CEH).

d. ATCTD - Along-Track-Cross-Track Distance Module:

(1) ATCD computes the along-track and cross-track distance (ATD and CTD) navigation parameters. It is called by the navigation executive during each navigation cycle.

(2) The ATD and CTD parameters are described as follows: the straight line between the previous waypoint and the destination waypoint defines the trackline; and the ATD describes how far the vessel would be from the destination waypoint if it were on the trackline; and CTD describes how far the vessel is from the trackline.

(3) ATD and CTD are computed in the following manner. Based on the received loran time differences (TD's) and the known time differences at the destination waypoint, the PILOT system, using a TD-to-XY transformation algorithm, calculates the vessel's X and Y coordinates referenced to the loran chain coordinate system.

(4) Since the waypoint coordinates are known, the vessel-to-waypoint distance (or range) can be calculated using the standard distance formula. The trackline makes an angle with the north line called the "destination bearing angle" (DBA). The vessel-to-waypoint line makes a second angle with the north line called the "present position bearing angle" (PPBA). If the vessel is on the trackline, then the two angles are equal. In this case the ATD is the same as the distance from the vessel to the waypoint, and the CTD is zero. If the vessel is not on the trackline, then a differential angle, PPBA-DBA, can be computed which describes by how much the vessel-to-waypoint line is to the right or left of the trackline and is used to give sense (left or right) to the CTD value.

(5) Thus, by computing the vessel-to-waypoint range and the differential angle, ATD and CTD can be found by trigonometric relationship found in the right triangle where ATD and CTD are the two sides and the range is the hypotenuse. ATD and CTD are computed by:

ATD = RANGE*COS(PPBA-DBA)
CTD = RANGE*SIN(PPBA-DBA)

ATD and CTD are in units of floating-point yards, since the coordinates are in these units. Conversions to other units are performed in the CATCT module (see Section 3.6.10(g)). Figure 3-25 illustrates the geometry of the along-track and cross-track distance (ATD and CTD) computations.

e. ATCTS - Along-Track/Cross-Track Speed Module:

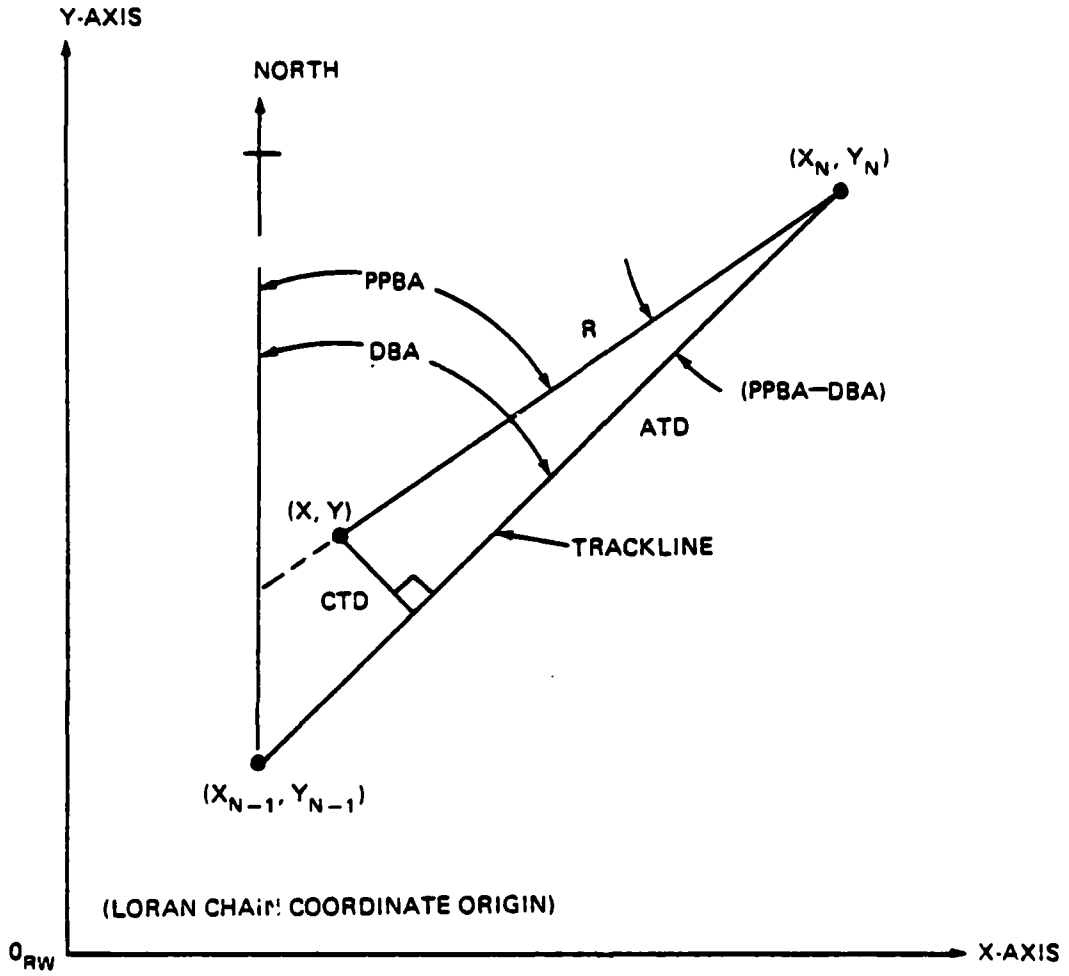
(1) ATCTS computes the along-track and cross-track speed (ATS and CTS) navigation parameters. It is called by the navigation executive during each navigation cycle.

(2) The ATS and CTS parameters are described as follows. The straight line between the previous waypoint and the destination waypoint defines the trackline. The vessel's velocity vector, V, can be resolved into two components with respect to the trackline: (1) a velocity component along the trackline and (2) a velocity component normal to the trackline. ATS is the magnitude of the velocity component along the trackline, and CTS is the magnitude of the velocity component normal to the trackline.

(3) ATS and CTS are computed in the following manner. The velocity is computed as stated in Section 3.6.11. The velocity vector is formed by the vector sum of the X and Y velocity components. The vessel speed is found as the magnitude of the velocity vector. The vector direction is the angle which the vector makes with the north line. This angle is called "course made good" (CMG). The trackline makes another angle with the north line called the "destination bearing angle" (DBA). If the vector direction were parallel to the trackline, then the two angles would be equal. In this case the ATS is the same as the vessel's speed, and the CTS is zero; i.e., there would be no component of velocity away from the trackline or a line parallel to it passing through the vessel. If the vector direction is not parallel to the trackline, then a differential angle, CMG-DBA, can be computed which describes by how much the velocity vector must be rotated to coincide with the trackline or a line parallel to it. The sign of the difference angle describes whether the vector points to the right or left of the trackline and is used to give sense (left or right) to the cross-track speed value.

(4) The velocity vector can be resolved into two components with respect to the trackline: (1) a velocity component along the trackline or a line parallel to it and (2) a velocity component normal to the trackline. ATS and CTS are the magnitudes of these two velocity components. ATS and CTS are computed by:

ATS = SPEED*COS(SMG-DBA)
CTS = SPEED*SIN(CMG-DBA)



(X, Y): VESSEL'S PRESENT POSITION COORDINATES

(X_N, Y_N): DESTINATION WAYPOINT (N) COORDINATES

(X_{N-1}, Y_{N-1}): PREVIOUS WAYPOINT (N—1) COORDINATES

PPBA: PRESENT POSITION BEARING ANGLE = $TAN^{-1}((X_N - X)/(Y_N - Y))$

DBA: DESTINATION BEARING ANGLE

R: RANGE, VESSEL-TO-DESTINATION WAYPOINT

$= ((X - X_N)^2 + (Y - Y_N)^2)^{1/2}$

DIFFERENTIAL ANGLE = PPBA — DBA

ALONG-TRACK DISTANCE: ATD = R*COS(PPBA—DBA)

CROSS-TRACK DISTANCE: CTD = R*SIN(PPBA—DBA)

FIGURE 3-25. ALONG-TRACK/CROSS-TRACK DISTANCE MODULE.

ATS and CTS are in units of floating-point yards per second, since the X and Y velocity components are in these units. Conversions to other units are performed in the CATCT module (see Section 3.6.10.(g)). Figure 3-26 illustrates the geometry for the along-track and cross-track speed (ATS and CTS) computations.

f. TIMTGO - Time to-Go Module:

(1) Routine TIMTGO computes the time-to-go (TTG) navigation parameter. It is called by the navigation executive during each navigation cycle.

(2) The TTG parameter describes approximately how long it will take the vessel to travel from its current position to the destination waypoint if the along-track speed and heading are maintained. TTG is computed by dividing the current along-track distance in yards by the current along-track speed in yards to determine the time-to-go in seconds:

$$TTG(SECS) = ATD(YDS)/ATS(YDS/SEC)$$

TTG is in units of floating-point seconds. The CATCT module performs the conversion of TTG from seconds to other units of measure.

g. CATCT - Along-Track/Cross-Track Parameter Conversion:

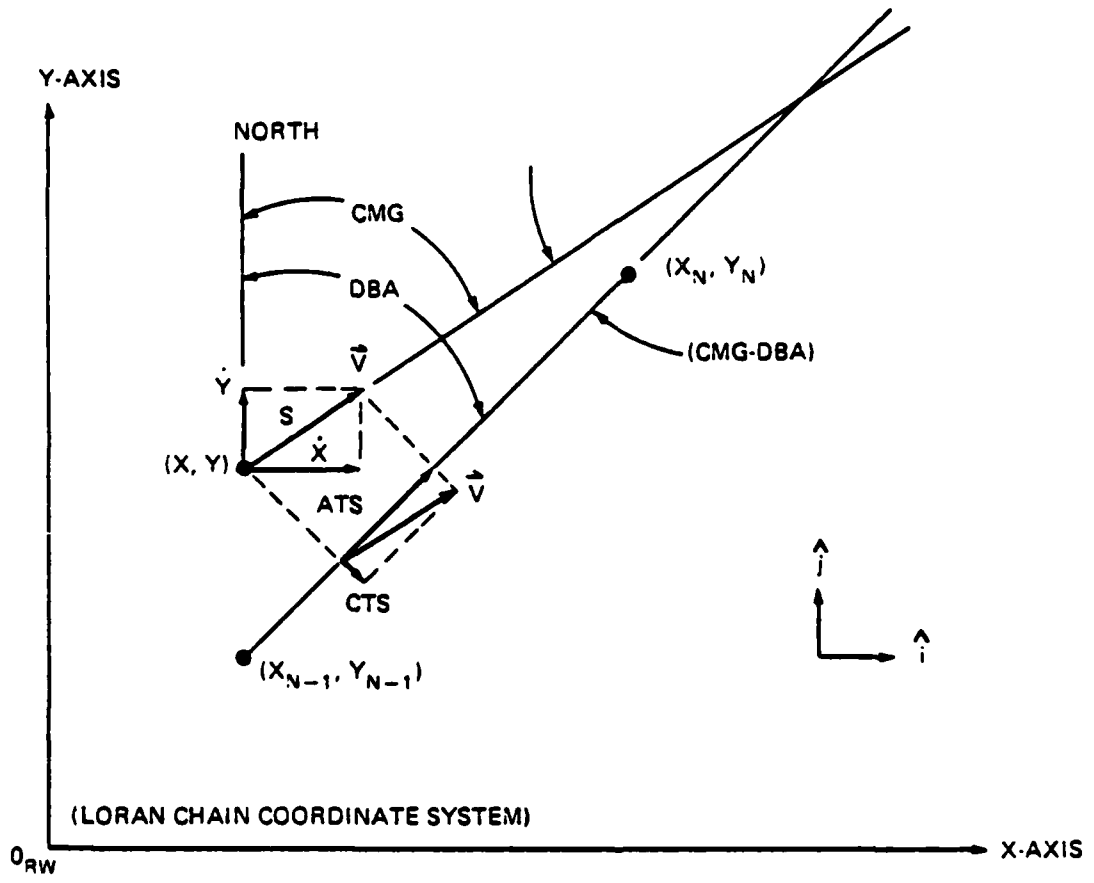
(1) Routine CATCT converts the navigation parameters from an internal set of units to the units required by the operator. The navigation executive calls CATCT during each navigation cycle.

(2) The vessel's position is computed as X and Y coordinates in a loran chain coordinate system. The coordinates are in units of floating-point yards. The velocity is computed as X and Y components given in yards per second. Therefore, the along-track and cross-track parameters are computed in yards and yards per second. However, the operator requires that these parameters be in more suitable units on the display; e.g., yards or miles, miles per hour, or minutes.

(3) CATCT converts the parameters in yards and yards per second to display units suitable for the operator. Since the along-track distance parameter can be displayed in yards or miles, CATCT determines, depending upon the magnitude, which unit of measurement is appropriate and sets a flag.

h. Position, Cross-Track Module - PCT:

(1) PCT determines whether the vessel is to the left or right of the trackline. It is called by the along-track, cross-track distance module, ATCTD, which is called by the navigation executive during each navigation cycle and computes a differential angle.



(X, Y): VESSEL'S PRESENT POSITION COORDINATES

(X_N, Y_N): DESTINATION WAYPOINT (N) COORDINATES

(X_{N-1}, Y_{N-1}): PREVIOUS WAYPOINT (N−1) COORDINATES

$(\dot{X}, \dot{Y})$: VESSEL'S X AND Y VELOCITY COMPONENTS

$\vec{V}$: VESSEL'S VELOCITY VECTOR

S: VESSEL'S SPEED = $((\dot{X})^2 + (\dot{Y})^2)^{1/2}$

CMG: "COURSE-MADE-GOOD" = $TAN^{-1}(\dot{X}/\dot{Y})$

DBA: DESTINATION BEARING ANGLE

DIFFERENTIAL ANGLE = CMG−DBA

ALONG-TRACK SPEED: ATS = S•COS(CMG−DBA)

CROSS-TRACK SPEED: CTS = S•SIN(CMG−DBA)

FIGURE 3-26. ALONG-TRACK/CROSS-TRACK SPEED COMPUTATIONS.

(2) The differential angle is computed by subtracting the destination bearing angle (DBA) from the present position bearing angle (PPBA). These angles are described in Section 3.6.10. The sign of the differential angle determines whether the vessel is to the left or right of the trackline. PCT evaluates the sign of the differential angle, PPBA-DBA, and indicates on the alphanumeric display whether the trackline is to the left or right of the vessel. The position indicator appears in the CTD alphanumeric line and will be an L or R following the numerical value for CTD.

i. Direction, Cross-Track Module - DCT:

(1) DCT determines the direction in which the vessel is moving, left or right, relative to the trackline, and puts this information on the alphanumeric display on the CRT. It is called by the ATCTS module, which is called by the navigaton executive during each navigation cycle and computes a differential angle.

(2) The differential angle is computed by subtracting the destination bearing, DBA, from the velocity vector heading angle called "course-made-good" (CMG). Course-made-good is defined as the angle which the velocity vector makes with a northline and is computed using the X and Y velocity components of the vessel. The angles, CMG and DBA, are described in Section 3.6.10. The sign of the differential angle tells whether the velocity vector points toward the right or left of the trackline. When it is resolved into components parallel to and normal to the trackline, the direction of the normal component points in the cross-track direction in which the vessel is moving, i.e., toward the right or the left. DCT evaluates the sign of the differential angle, CMG-DBA, and indicates on the alphanumeric display whether the vessel is moving in the right or left cross-track direction. The direction indicator appears in the CTS alphanumeric line and will be an ASCII L or R following the numerical value for CTS.

3.6.11 Velocity Calculation - VC: The velocity calculation (VC) used by PILOT is a first-order filter based on a change in position. The equation used is:

$$V = V\,(i-K) + K\,\frac{(D-D_o)}{\Delta t}$$

where

V	=	velocity
K	=	constant less than 1
D	=	current distance
D_o	=	previous distance
Δt	=	change in time

VC uses this equation to compute an X and Y component of velocity.

3.6.12 Switching Control Module - SWI:

a. Routine SWI determines when it is necessary to bring a new file or files from the cartridge tape and switches files before, at, and after the waypoint. It is called by the navigation executive during every navigation cycle.

b. A determination is made by SWI if the vessel has crossed along-track switchpoints and sets flags accordingly. The switchpoints are stored on the cartridge tape and are brought in when a file is read in. The tape contains two types of switchpoints: the master file switchpoint and the detail file switchpoint. SWI compares the magnitude of the completed along-track distance values with these switchpoints. The sign of the computed along-track distance is also evaluated. Each master file contains one or more detail files. If the vessel has crossed a detail switchpoint, but not the master switchpoint, then a flag is set that directs the navigation executive to read in the next detail file. If the master switchpoint has been crossed, then a flag is set that directs the executive to read in the next master file and its first detail file. If along-track distance is negative, then the vessel has crossed over a waypoint and is leaving it. SWI adjusts the alpha-numeric display accordingly, telling the operator that it is leaving the waypoint at some new bearing angle. This angle is the bearing to the next waypoint.

c. Routine SWI accepts as input the switch status flag, SWIF, a single-byte variable stored in HCOM, and action is taken according to the information contained in SWIF. SWI also accepts as input the detail status flag, DSTAT+1, also stored in HCOM. DSTAT+1 is examined to determine if the present detail file is also the last one for the present master file.

d. The output provided by SWI is an updated switch status flag, SWIF, based on which action was taken. The navigation executive examines the switch status flag to determine whether it needs to read a new file or files and, if so, which type or types of files.

3.6.13 Line Printer and Tape Output:

3.6.13.1 Line Printer/Tape Dump Module - LPTOUT:

a. LPTOUT is the module called by the PILOT executive whenever output to the line printer or tape is enabled.

b. LPTOUT makes use of the following utility routines: PROUTH, PROUT, PRREC, PCRLF, WAR, and WAEOF.

3.6.13.2 Header Module - PROUTH: PROUTH outputs the header record to the PILOT print buffer PRBUF whenever it is called. The header record consists of column headings in one line and the column units in the next line.

3.6.13.3 Data Output Module - PROUT:

a. PROUT outputs the data record to the PILOT print buffer PRBUF whenever it is called.

b. The data record consists of 136 bytes of ASCII data representing the variables to be output to the printer or tape. PROUT builds up this data record by converting each variable from its internal storage format into its equivalent ASCII string and storing it in an appropriate location within the print buffer.

3.6.13.4 Print a Record Module - PRREC:

a. PRREC is used to send a record of data in the print buffer PRBUF to the HP 2631G printer. The printer is used to log PILOT data in a human readable form. The printer must be "on line" if PRREC is to operate correctly.

b. The HP 2649A terminal's subroutine PUTIO is used by PRREC and performs all required setup tasks. PUTIO is a general purpose utility subroutine that can transfer a record of data to a selected device. PRREC receives as input parameters the starting address of the data record and the number of bytes in the data record. PUTIO uses two buffers (IOBUF1 and IOBUF2) which are 256 bytes long. The maximum byte count passed to PRREC in the B register is 255. PRREC selects the HP 2631G printer as the receiving device, gets one of the buffers, and then calls PUTIO. PUTIO transfers the data in its buffer to the printer. PRREC then frees the PUTIO buffer before returning to the caller.

c. PRREC assumes ASCII data in the data record to be output to the printer. Other data formats have no meaning on the printer listing. Since PRREC does not issue a carriage return and line feed to the printer, the PCRLF module accomplishes these functions (see Section 3.6.13.5).

3.6.13.5 Printer Carriage Return/Line Feed Module - PCRLF:

a. PCRLF is a printer utility module used to perform a carriage return and line feed on the HP 2631G printer. The printer is used as a data logger when the PILOT system is operating in a navigation mode.

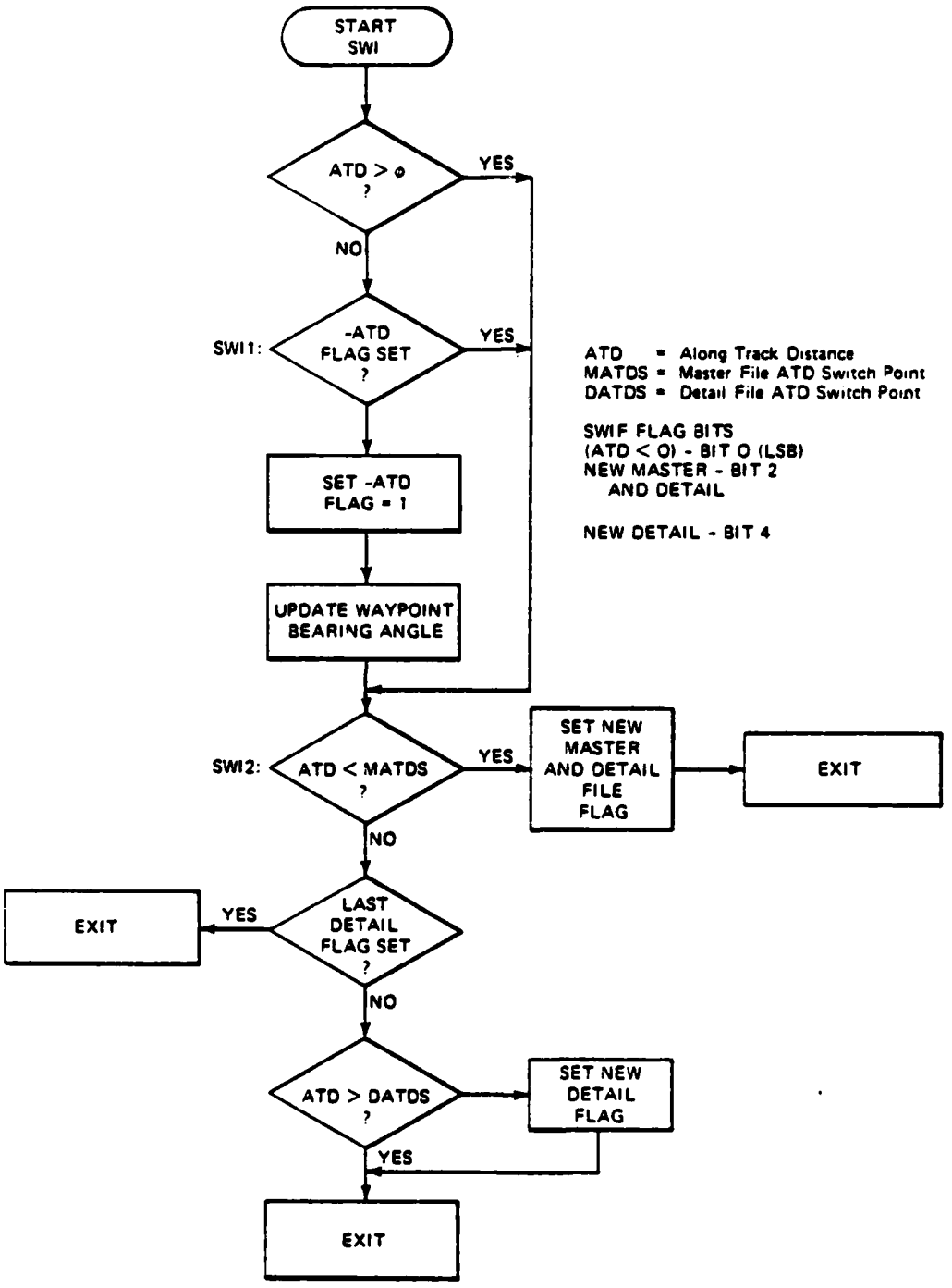


FIGURE 3-27. SWITCHING CONTROL MODULE - SWI.

b. PCRLF sets up ASCBUF with ASCII carriage return and line feed characters and then calls the utility subroutine PRREC. PRREC sends the data to the printer which causes it to perform a carriage return and a line feed.

3.6.13.6 Tape Write Routine - WAR:

a. WAR is called to transfer print buffer data to the right tape drive of the PILOT terminal. A tape must be installed in the right drive for this routine to operate properly.

b. Data written on tape has the same format as that written or printed by the line printer. The data are stored as ASCII characters.

3.6.13.7 Tape Write End of File - WAEOF: WAEOF is called to write end-of-file (EOF) data on the right tape drive. EOF marks are used to separate each record written on the tape.

3.6.14 Vessel Symbol:

3.6.14.1 Vessel Scaling - VESDIM: The vessel symbol displayed by PILOT, when using the detail chartlet, is scaled to the dimensions of the vessel on which PILOT is installed. VESDIM performs the scaling operation by reading the three binary switches located on the TD Bias and Heading Board, and converts each of these switch readings into floating-point math pack numbers for later use. The operator sets the vessel length, vessel width, and location of the antenna from the bow. Each binary switch consists of eight toggle switches. The value of each toggle is as follows:

Toggle No.	Length and Antenna (in feet)	Beam (in feet)
1	512	128
2	256	64
3	128	32
4	64	16
5	32	8
6	16	4
7	8	2
8	4	1

3.6.14.2 Simple Vessel Symbol - VESSIM: When a master picture is being displayed, VESSIM draws a cross hair on the CRT at the point representing the vessel's position. The cross hair consists of a half-inch vertical line intersecting a half-inch horizontal line. The intersection defines the vessel's position. The cross hair is drawn in the complement video mode defined by the HP 2649A terminal. It can be erased by redrawing it in the same mode. Other graphics remain undisturbed when using the complement video feature. The navigation executive determines if a master picture is being displayed, and, if so, will call the VESSIM subroutine.

3.6.14.3 Detail Vessel Graphic Software:

a. In contrast to the simple cross hair employed for a master picture, the vessel graphics software required for a detail picture is considerably more involved.

b. The detail vessel graphics software performs the following functions:

1. For each new detail file it scales the vessel to the CRT using the vessel length, width, and antenna position as input parameters.

2. For each navigation cycle it translates the vessel symbol to its proper location on the CRT, rotates the vessel symbol to align it with the vessel's heading, and draws the vessel symbol on the CRT.

c. The vessel graphics software consists of three modules:

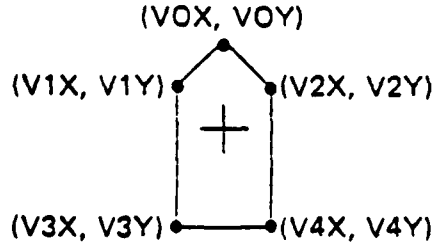
1. VESSCL: Scales the vessel symbol to the CRT for each new detail file (when the scale factor can change).

2. VESSYM: Translates and rotates the vessel symbol to its proper position on the CRT.

3. VESDRW: Draws the vessel symbol in complement video mode on the CRT.

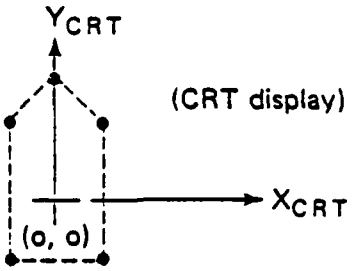
3.6.14.4 Vessel Symbol Scaling Module - VESSCL:

a. VESSCL is one of three modules which constitute the detail vessel symbol software. This module scales the vessel symbol to the display, using the detail scale factors of the vessel length, width, and antenna position as input parameters. It is called by the navigation executive whenever a new detail file is read from the cartridge tape. VESSCL generates as outputs five coordinate pairs of numbers. These five pairs of numbers define five points representing the vertices of an irregular pentagonal figure as shown below:



This pentagonal figure defines the archetypical vessel symbol. The parameter, VESLEN, determines the symbol length; the parameter, VESWID, the width, and the parameter, ANTENA, the antenna position. These parameters are fixed and are set at the time of installation. The plus sign (+) identifies the antenna position.

b. VESSCL computes the X and Y coordinates of the five points relative to the antenna position. The antenna position coincides with the display origin, i.e., the lower left-hand corner of the CRT screen. The CRT coordinate system is defined so that the Y-axis coincides with the vessel symbol longitudinal axis, and the X-axis intersects it at the antenna position marking the origin as illustrated below:



The coordinates of the five points are computed and stored as 32-bit, floating-point numbers. The conversion to CRT coordinates (2's complement, 16-bit binary integers) occurs in module, VESSYM. The five points are identified as ordered pairs: (V0X, V0Y), (V1X, V1Y), (V2X, V2Y), (V3X, V3Y), and (V4X, V4Y).

3.6.14.5 Vessel Symbol Module - VESSYM:

a. VESSYM is one of three modules constituting the detail vessel symbol software. This module translates and rotates the vessel symbol model computed in VESSCL. Whenever the displayed picture is a detail, VESSYM is executed once per navigation cycle. It is called by the navigation executive.

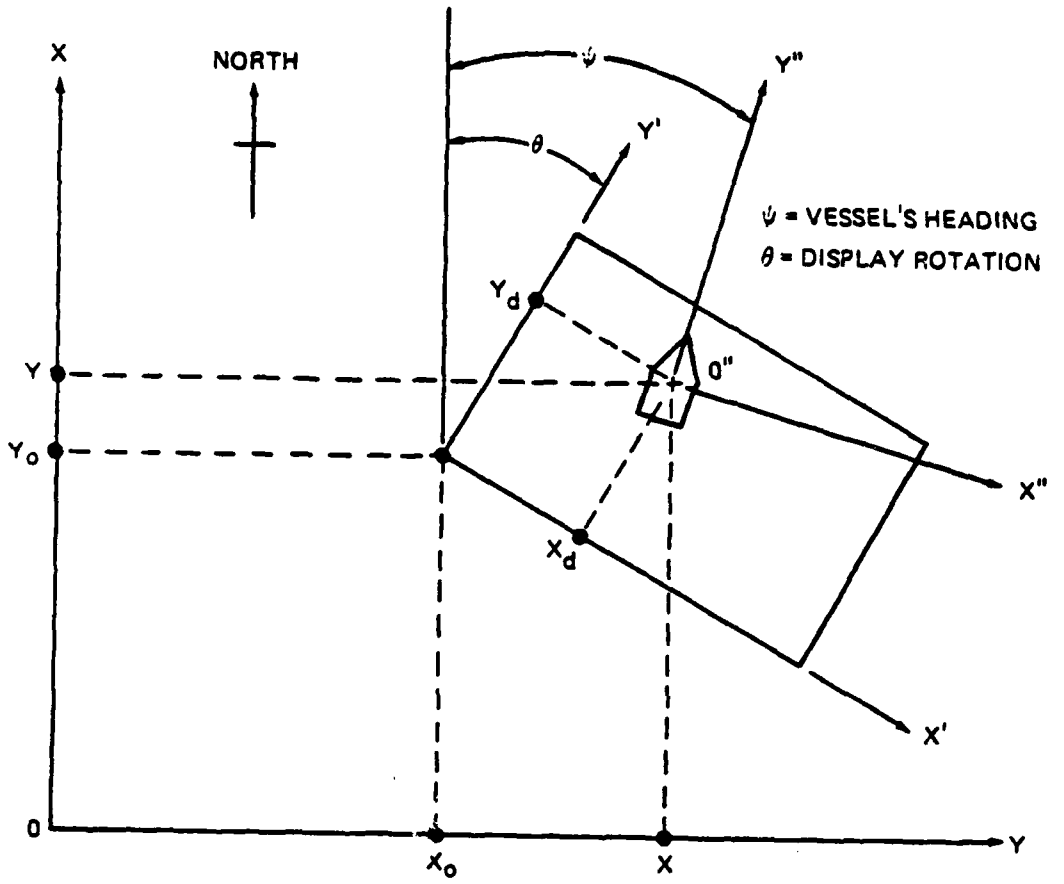
b. To provide a graphical output to the operator, the model must be translated and rotated with respect to the display origin, which coincides with the CRT screen origin (the lower-left corner) (see Fig. 3-28).

c. The inputs to VESSYM include the model coordinates computed in VESSCL and the current vessel position X,Y and heading.

d. The output of VESSYM consists of a set of five ordered pairs. These five coordinate pairs represent the actual vessel symbol to be drawn on the CT, and are designated as: (BX, BY), (LBX, LBY), (RBX, RBY), (LSX, LSY), and (RSX, RSY). These variables are 2's complement, 16-bit binary integers and are recomputed each time VESSYM is executed.

3.6.14.6 Vessel Draw Module - VESDRW:

a. VESDRW is one of three modules constituting the detail vessel symbol software. This module draws the vessel symbol on the CRT in the complement video mode. This mode, supplied as part of the HP terminal, allows graphics images to be drawn in such a way that, when erased, other graphics images are undisturbed. Thus, the previous vessel symbol is erased by drawing it again in the complement video mode which leaves the background graphics intact. The previous symbol is erased before new coordinates are computed for the current symbol.



3 COORDINATE SYSTEMS:

1. X, Y: REAL WORLD (LORAN CHAIN) COORDINATE SYSTEM
2. X', Y': DISPLAY COORDINATE SYSTEM, ROTATED BY ANGLE θ WITH RESPECT TO X, Y
3. X", Y": VESSEL COORDINATE SYSTEM, ROTATED BY ANGLE ψ (HEADING) WRT X, Y

POINTS WITHIN COORDINATE SYSTEMS:

1. (X, Y): VESSEL'S POSITION WRT X, Y
2. (X_o, Y_o): DISPLAY ORIGIN WRT X, Y
3. (X_d, Y_d): VESSEL ROTATION WRT X', Y'

RELATIVE ROTATION:

($\psi-\theta$): VESSEL ROTATION WRT X', Y'

FIGURE 3-28. VESSYM-COORDINATE SYSTEMS AND ROTATIONS.

b. Inputs to VESDRW are the five pairs of CRT coordinates calculated by VESSYM (see Section 3.6.14.5). It outputs the vessel symbol image by drawing solid vectors in the graphics memory connecting the five pairs of points.

c. During one navigation cycle, VESDRW is called twice by the navigation executive: to erase the previous symbol and again after a new symbol has been computed by the VESSYM module.

3.6.15 CTD BAR:

3.6.15.1 Bar Graph Module - BG:

a. BG displays the bar graph and the vessel image whenever the operator has requested CTD BAR via the keyboard. The BG module is called directly by the navigation executive whenever the bar graph display is wanted.

b. The bar graph display shows a calibrated distance scale and the corresponding labels at the bottom of the CRT screen. Two scales are currently implemented: ±100 yards and ±500 yards. Angle brackets are positioned according to channel width data from the cartridge tape. The zero position on both scales falls at the middle of the CRT screen. Superimposed on the graph and presented in inverse video is a scaled vessel image field. This field is positioned on the graph by using the vessel's position relative to the desired trackline. The bar graph display gives the operator another representation of his vessel in the waterway with respect to a trackline.

c. The bar graph software consists of four modules: BG, BGON, BLON, and BGOFF. BG acts as the overall control module for the bar graph software. BGON displays the bar graph, i.e., it outputs a CRT line of plus (+) signs that represent an analog distance scale. BLON puts one of two sets of bar graph labels on the display. BGOFF turns off the bar graph and the labels, and it can be called directly by the navigation executive.

d. BG performs the following sequence of operations. First, BG erases the previous bar graph display to wipe out memory of the previous inverse video field. After putting up a new bar, it selects one of two sets of labels based on the magnitude of CTD and labels the bar graph. Then, based on the magnitude and the sign of CTD, it calculates the center of the vessel image field within the bar graph. Using the vessel center, BG calculates the starting and terminating points for the inverse video vessel field. The field width is calculated from the vessel width, VESWID, and the bar graph scale. VESWID is assumed to be in floating-point yards.

e. BG accepts as an input the value for cross-track distance (CTD). CTD is assumed to be in units of yards and in floating-point form. The output consists of the bar graph display. The bar graph is represented by two lines in the alphanumeric display memory. The first line contains ASCII plus (+) signs representing calibrated tick marks on a distance scale. The second line contains the labels for the bar graph. The vessel image is represented by an inverse video field placed into the first line. When displayed on the CRT screen, the first line appears in row 22 and the second line appears in row 23.

3.6.15.2 Bar Graph on Module - BGON:

a. BGON is one of four modules constituting the bar graph display software. When the operator has requested CTD BAR via the keyboard, the navigation executive will call module BG. BG, in turn, calls BGON to turn on the bar graph.

b. The bar graph is represented by one line of alphanumeric display memory. The data contained in this line are ASCII plus (+) signs. The plus signs represent tick marks of a calibrated scale. When displayed on the CRT screen, the bar graph appears as nine evenly spaced tick marks in row 22, starting in column 3.

c. BGON uses the HP terminal's "character-insert" subroutine CHRINS.

3.6.15.3 Bar Labels on Module - BLON:

a. BLON is one of four modules which constitute the bar graph display software. When the operator has requested CTD BAR via the keyboard, the navigation executive will call module BG. BG, in turn, calls BLON to label the bar graph generated by module BGON. BG is the only module that can call BLON.

b. As currently implemented, the bar graph display can be one of two fixed scales: ±100 yards and ±500 yards. Accordingly, two sets of labels are defined. As described in Section 3.6.15.1, the bar graph consists of one line of alphanumeric display memory containing ASCII plus (+) signs which represent tick marks for a calibrated distance scale. In both cases the zero position of the bar graph falls in the center of the 80-character line. The normal bar graph labels (±100 yards scale) show tick marks as 25-yard increments to either side of the zero position. The expanded bar graph labels (±500 yards scale) show the tick marks as 125-yard increments to either side of the zero position. In both cases the labels to the left of the zero position do not have preceding minus (-) signs.

c. BLON has two entry points: BLON1 and BLON2. BLON1 is used for the normal scale (±100 yards) bar labels and BLON2 is used for the expanded scale (±500 yards) bar labels.

d. Whichever entry point is used, BLON performs the following operations. The selected set of labels is transferred to the print buffer, PRBUF. Then the print buffer is written into the alphanumeric display memory. When displayed on the CRT, the bar graph labels appear in row 23, starting in column 3, and are lined up directly below the tick marks.

3.6.15.4 Bar Graph Off Module - BGOFF:

a. BGOFF is one of four modules which constitute the bar graph display software. Its function is to turn off the bar graph display. It can be called by the BG module or by the navigation executive. The navigation executive will call BGOFF when the operator requests that the bar graph

display be turned off. When the operator has requested the bar graph via the CTD BAR key on the keyboard, the navigation executive will call the BG module. BG also calls BGOFF.

b. When the CTD BAR is on, an inverse video field representing the vessel appears in the bar graph. The placement and width of this inverse video field depends on the vessel's position with respect to the trackline, the vessel's width, and the bar graph scale. It is calculated by the BG module. However, the HP 2649A terminal is configured such that it "remembers" where the inverse video field is. Consequently, it is necessary to "clear" the memory of the previous inverse video field before attempting to create a new one. BGOFF performs this function by deleting from memory the alphanumeric line containing the inverse video field.

c. BGOFF uses the HP terminal's "clear screen" subroutine CLEARS. The alphanumeric cursor is positioned at the start of row 22 prior to calling CLEARS.

3.6.16 Range and Bearing - RAB:

a. The software for the range and bearing function is contained in subroutine RAB, which contains four entry points: IRAB, RAB, ERAB, and IRABL.

b. The range and bearing function is activated by the following key sequence: RANGE & BEARING key, move graphics cursor to desired destination point, and EXECUTE key. After this sequence has been completed, the PILOT executive calls IRAB to initialize the range and bearing function. IRAB saves the destination point coordinates, draws the destination symbol, writes the display labels, and sets the range and bearing flag on.

c. While the range and bearing flag is on, the PILOT executive calls RAB. RAB computes the range and bearing from the vessel to the destination point, and displays the data on the screen.

d. The range and bearing function is terminated by the following key sequence: CLEAR, RANGE and BEARING, and EXECUTE. After this key sequence, the PILOT executive calls ERAB to terminate the range and bearing function. ERAB will set the range and bearing flag off, erase the destination symbol, and erase the range and bearing alphanumeric data on the screen. The executive also terminates the range and bearing function when master/detail switching occurs, or when the next tape chartlet is read.

e. Entry point IRABL is called by the executive to write the range and bearing labels on the screen. It is called whenever the HP terminal alphanumeric display is cleared while range and bearing is on.

f. Module FRAM defines range and bearing variables which are located in the fast RAM area. Fast RAM space is available from locations 90E0 H to 90FF H.

3.6.17 Project Mode:

a. The project mode displays a predicted position of the vessel a short time in the future. Three types of projections are available: a linear projection based on the vessel's course-made-good (CMG), a curved projection based on the vessel's CMG and turning rate, and a linear projection based along the vessel's heading.

b. The type of project is selected by changing the project variable (PJV) in the status menu. All projections are made for a fixed amount of time in the future. The project time is entered after activating the project mode.

c. The project mode is activated by the following key sequence: PROJECT key, enter project time in minutes, and EXECUTE key. After this sequence has been completed, the PILOT executive will call subroutine PROJI to initialize the project software. PROJI converts the entered project time from ASCII to floating point, and also computes the project interval time (PJTC) from the project time in seconds divided by the number of segments (8) in the project line.

d. If no project time is entered, then a default time of two minutes is used. If zero time is entered, then the project mode is terminated. The project flag (PJF) on-off bit is set on. The first time through bit, which indicates that a project line has been drawn on the display, is not changed. If the project mode is already on, a new project time can be entered by repeating the above key sequence.

e. The PILOT executive calls subroutine PROJ while the project flag is on. PROJ is the driver subroutine that computes the project line X and Y coordinates and draws the project line on the display. There are subroutines to compute the project line CRT coordinates for each of the three project types. PROJ checks the project variable (PJV) and calls the subroutine for the selected project type, as described in the following sections.

3.6.17.1 Subroutine PROJ1: PROJ1 computes the project line coordinates for the course-made-good (CMG) linear project type (PJV=1). The X and Y CRT coordinates for the project line are computed as follows:

For k = 0 to 7
X(k+1) = X(k) + VX * PJTC
Y(k+1) = Y(k) + VY * PJTC

where: VX = X component of ship's velocity
VY = Y component of ship's velocity
PJTC = project interval time

The computed coordinates are saved in buffer PJXY.

3.6.17.2 Subroutine PROJ2: PROJ2 computes the project line coordinates for the CMG curved project type (PJV = 2). The equations for the X and Y coordinates have been derived so that they will yield a finite result when the turning rate is zero. The project software uses the following equations:

$$\Delta X(k) = VO * PJTC \left[\frac{\Delta\Psi}{2} * \cos \Psi 0 + \sin \Psi 0\right]$$
$$\Delta Y(k) = VO * PJTC \left[\cos \Psi 0 - \frac{\Delta\Psi}{2} * \sin \Psi 0\right]$$

where:

VO = ship's speed
PJTC = project interval time
$\Delta\Psi$ = turning rate * project interval time
$\Psi 0$ = present heading angle of ship.

The terms cos $\Psi 0$ and sin $\Psi 0$ are updated after every iteration. The ship's turning rate is computed from:

$$\dot{\Psi} 0 = an/VO$$

where an is the ship's radial acceleration as output from the heading filter. The computed coordinates are saved in buffer PJXY.

3.6.17.3 Subroutine PROJ3: PROJ3 computes the project line coordinates from the gyro linear project type (PJV = 3). This project type produces a linear project line along the ship's heading. PROJ3 first computes the velocity component along the ship's heading as follows:

$$VO' = VO * \cos(\pi/2 - HDG - CMG)$$

where:

VO = ship's speed
HDG = heading angle from gyro
CMG = ship's course made good

X and Y coordinates of this new velocity vector are computed, using the following equations:

$$VX' = VO' * \sin(HDG)$$
$$VY' = VO' * \cos(HDG)$$

PROJ3 then jumps to subroutine PROJ1, which uses these new X and Y velocity components to compute the project line coordinates.

3.6.17.4 Subroutine PROJL: PROJL is called by the PROJ driver to draw the project line on the display. The project line contains eight segments, and a solid line is drawn in complement video mode between alternate pairs of points. The X and Y coordinates of the eight project line points must be saved in buffer PJXY before calling PROJL. The project line coordinates appear in PJXY (32 bytes total) in the following order:

PJXY = X1MSB, X1LSB, Y1MSB, Y1LSB, X2MSB, X2LSB, Y2MSB,
Y2LSB, ... X8MSB, X8LSB, Y8MSB, Y8LSB.

Each coordinate is two bytes (single fixed-point format), four bytes per X and Y coordinate pair. The data in PJXY is saved until the next time PROJ is called, so that the previous project line can by erased by calling PROJL before new coordinates are computed.

3.6.17.5 Subroutine PROJX:

a. PROJX is called to end the project mode. PROJX will erase the previous project line, set the project on-off bit to off, and clear the first time through bit. The project mode is terminated by the following key sequence: CLEAR, PROJECT, and EXECUTE. The PILOT executive calls PROJX after this key sequence to end the project mode. If the project mode is already off when PROJX is called, then the bell rings and no action is taken.

b. The project mode can be ended by entering the above key sequence, or by entering a zero project time. Project does not end on a master/detail switch or at a waypoint file switch. The first time thru bit in the project flag (PJF) is cleared on a master/detail or waypoint file switch so that PROJ does not try to erase the old project line. The old project line is erased when the display is cleared by the switch software.

3.6.18 Statistics:

3.6.18.1 Statistical Package:

a. The PILOT statistical package computes a mean and standard deviation for TDA, TDB, TDC, X, and Y based on 25 samples. The mean and standard deviation for each quantity is printed on the line printer.

b. STAT is the main statistical module. The entry point STAT performs the setup required for the statistical package. Entry point STATU updates each variable based on a new raw data point. The module UPDATE performs the updating of each variable. The module STATS1 calculates the mean and standard derivation for each variable, PRSTHD prints the statistical header, and PRSTAT prints each line of statistical data. These modules are described in greater detail in the following sections.

3.6.18.2 STAT Module: STAT is the main module in the statistical package. STAT performs the required initialization, the updating of each variable using the module UPDATE, computes means and standard deviation of each variable using STATS1, and prints the results on the line printer. STAT is called whenever the statistical flag, SSF, is set.

3.6.18.3 UPDATE Module: UPDATE performs the updating of each variable according to the following equation:

$$Z = x_i - x_{REF}, \Sigma Z, \Sigma Z^2$$

3.6.18.4 STATS1 Module: STATS1 computes the mean and standard deviation of each variable, using the following equations:

$$\bar{X} = \frac{1}{n} \Sigma Z + X_{REF}$$

$$S = \sqrt{\frac{\Sigma Z^2 - \frac{(\Sigma Z)^2}{N}}{N-1}}$$

3.6.18.5 PRSTHD Module: PRSTHD prints the statistical header.

3.6.18.6 PRSTAT Module: PRSTAT prints a line of statistical data. On entry RARAM points to the paramater name; Z MEAN points to the mean value; and Z STDV points to the standard deviation. PRSTAT converts the numbers to ASCII and prints them.

3.6.19 Utilities:

3.6.19.1 Conversion Routines:

a. ASCII Floating-Point Conversion Routines: Routines ASCFP and FPASC provide a means for generating in RAM the AMD double-precision floating-point numbers from ASCII strings in RAM (ASCFP), and reproducing ASCII strings from AMD double-precision floating-point numbers (FPASC). Routines ASCPF and FPASC are described in the following sections.

b. Routine ASCFP: ASCFP will convert an ASCII string of up to 8 characters to an AMD double-precision floating-point number. The address of the ASCII string (least numeric address and most significant character) is in the DE register. The HL register contains the address of the AMD "Math Pack" double-precision floating-point number in standard PILOT address form (least numeric address and most significant byte - THE EXPONENT). ASCII input strings may be integer or fixed point, with a terminating non-numeric character. If the sign is omitted, a plus (+) is assumed.

c. Routine FPASC: FPASC converts an AMD double-precision, floating-point number to an 8-character ASCII string. The address of the ASCII string (least numeric address and most significant character) is in the DE register. The HL register contains the address of the AMD "Math Pack" double-precision floating-point number in standard PILOT address form (least numeric address and most significant byte - THE EXPONENT). The ASCII strings produced are left justified.

3.6.19.2 Routines ASCDF and DFASC:

a. Routine ASCDF: ASCDF will convert an 8-character ASCII to double-precision, fixed-point math pack form. ASCDF will convert only positive integers from ASCII to BCD, and then a BCD binary routine (BCDBIN) converts the number to double-precision, fixed-point form.

b. Routine DFASC: DFASC performs the inverse function of ASCDF, but will only convert positive numbers in the range 0_{10} to $99,999,999_{10}$.

3.6.19.3 Degress to Radians Conversion - RDNDEG and DEGRDN:

a. The utility subroutine RDNDEG converts a number in radians into degrees by multiplying by a factor of 180/π. The numbers in radians are assumed to be in math pack floating-point notation. RDNDEG is located in the NUCR2 module.

b. Subroutine DEGRDN performs the reverse function of RDNDEG by converting a floating point number in degrees into radians by multiplying by a factor of π/180. DEGRDN is located in the UTIL module.

3.6.19.4 Angle Conversions - DEGRAD and RADDEG: The PILOT unit displays all angles using a navigational reference, where North is 0^{o} and positive angles are measured clockwise from North. Angles used in performing the calculations are expressed in trigonometric reference system, where East is 0^{o} and positive angles are measured counterclockwise from East. The math chip used in performing the calculations requires angles be expressed in radians rather than navigational degrees.

a. Subroutine DEGRAD converts a navigational degree measurement to trigonometric radians. The equation used to perform the conversion is: Radians = $\pi/180$ (90-degrees). The number to be converted is on the top of the math pack stack and the result is left on the top of the math pack stack.

b. Subroutine RADDEG converts from trigonometric radians to navigational degrees. The equation used to perform the conversion is: Degrees = 90 - ($180/\pi$ radians). The number to be converted is on the top of the math pack stack and the result is also on the top of the math pack stack.

3.6.19.5 Hypotenuse Calculation Module - HYPON: The utility module HYPON consists of the subroutines HYPON1 and HYPON2 which calculate the hypotenuse of a right triangle.

a. Subroutine HYPON1 within HYPON calculates the hypotenuse according to:

(P3) = SQRT [(P1)**2 + (P2)**2]

b. Subroutine HYPON2 also within HYPON calculates the hypotenuse according to:

(P5) = SQRT [(P1-P2)**2 + (P3-P4)**2]

c. P1, P2, P3, P4, and P5 are indirect address pointers defined in HCOM. In the first equation above, P1 and P2 point to the least significant bytes of two floating-point numbers representing the two sides of the triangle; P3 points to the most significant byte of where the floating point result is to be stored. In the second equation above, P1, P2, P3, and P4 point to the least significant bytes of four floating point numbers, and P5 points to the most significant bytes of where the floating point result is to be stored. HYPON1 and HYPON2 are the two entry points to HYPON.

3.6.19.6 Arctangent Computation Angle - ARCTAN: The utility module ARCTAN consists of the subroutines ATAN1 and ATAN2 which calculate the arctangent of a specified argument.

a. Subroutine ATAN1 calculates the arctangent of the ratio of two floating point numbers according to:

P3 = ARCTAN (P1/P2)

b. Subroutine ATAN2 calculates the arctangent of the ratio of two differences of floating point numbers according to:

P5 = ARCTAN [(P1-P2)/(P3-P4)]

c. P1, P2, P3, P4, and P5 are defined in HCOM and are indirect address pointers to the floating point numbers to be used. The resulting arctangent is a value between $-\pi/2$ and $+\pi/2$ floating point radians.

3.6.19.7 Trigonometric Function Utility - TRIG:

a. The utility module TRIG calculates an elementary trigonometric function of the form:

X = A*SIN (ARG)

or

X = A*COS (ARG)

where A is the amplitude and ARG is the argument. Because the sine and cosine operations work with floating point numbers, TRIG assumes the amplitude and argument to be floating point numbers.

b. TRIG uses indirect addressing to obtain its input and output parameters. It executes the following equation:

P3 = P1*(TRIG FCN (P2))

where TRIG FCN is the sine or cosine command in the accumulator. P1 contains the address of the least significant byte of the amplitude number, and P2 contains the address of the least significant byte of the argument or angle number. P3 contains the address of where the most significant byte of the floating point result is to be stored. P1, P2, P3, and the accumulator are set up prior to entry into TRIG.

3.6.19.8 Escape Sequence Interrupter - ESCI: The escape sequence command structure provides the operator with a powerful means of controlling the Hewlett-Packard terminal. Escape sequences are written and executed under complete software control. Routine ESCI provides the software necessary for executing an escape sequence which has been written by any software module. The only requirements are that a string be in ASCII, start with an escape character (1B H), and end with a null character (00 H). To call ESCI, place address of the string in the H and L registers.

3.6.19.9 Data Transfer - XFER: Utility program XFER transfers data from one area of memory to another. The source address of the data is specified in the DE register pair and the destination address of the data is specified in the HL register pair. The number of bytes to be moved is specified in the B register. On exit, B is zero and HL and DE are incremented by N.

3.6.19.10 Square of a Number - SQRX or FXSQRX: Utility SQRX calculates a square of a number which is on top of the math pack stack. SQRX calculates

the square of a floating point number. Utility FXSQRX calculates the space of a fixed-point number. In each case the result remains on top of the stack.

3.6.19.11 CRT to Real World Coordinate Transformation - CRTRW:

a. Utility CRTRW transforms coordinates between the CRT coordinate system and a real-world (loran chain) coordinate system.

b. The inputs to CRTRW are the CRT coordinates of some point on the screen. These coordinates are given as 16-bit, 2's complement binary integers stored at locations CRTX, CRTX+1 and CRTY, CRTY+1. Other inputs are the display origin coordinates, the display rotation angle, and the CRT scale factor. These inputs are stored as floating point numbers.

c. The outputs of CRTRW are real-world X and Y coordinates of the point in question. The real-world coordinates are 32-bit floating point numbers. The X coordinate is stored at the floating point variable XRW and the Y coordinate is stored at the floating point variable YRW.

d. Figure 3-29 illustrates the two coordinate systems and the transformation equations involved.

3.6.19.12 Real World to CRT Coordinate Transformation - RWCRT:

a. Utility RWCRT transforms coordinates between a real-world (loran chain) coordinate system and the CRT coordinate system.

b. The inputs to RWCRT are the coordinates of some point in the real-world coordinate system, the coordinates of the display origin, the display rotation angle, and the CRT scale factor. These quantities are stored as floating point numbers.

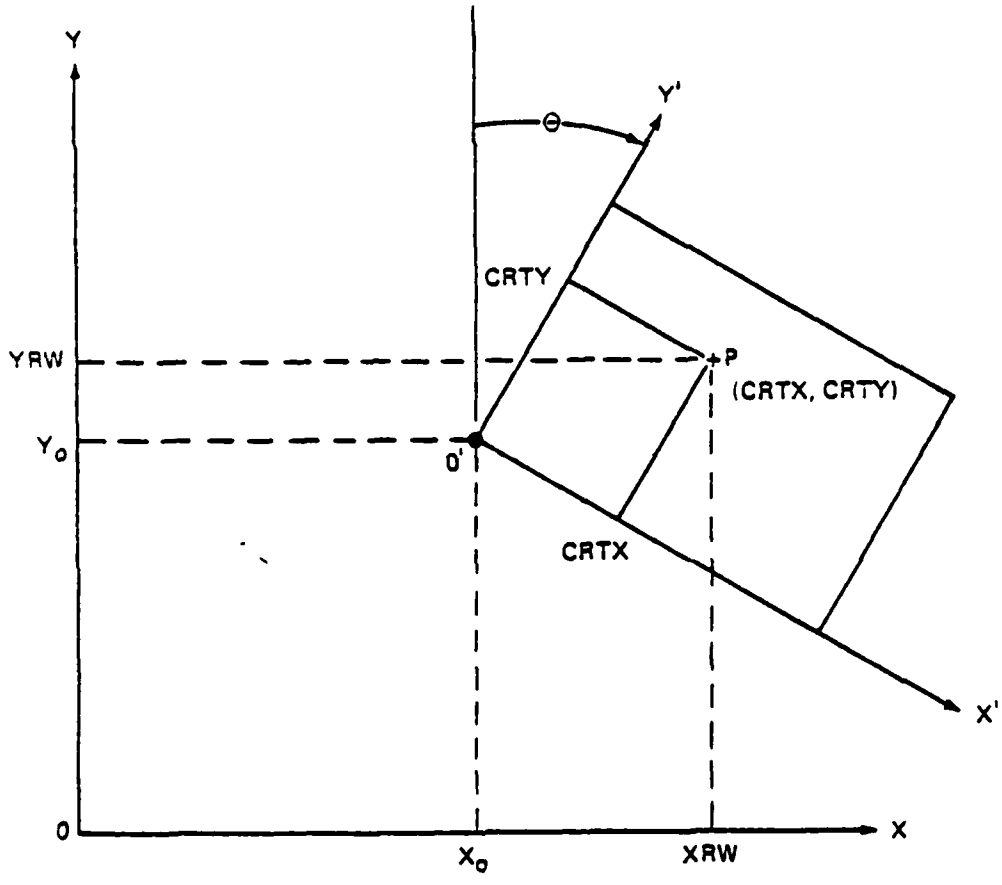
c. The outputs of RWCRT are the CRT X and Y coordinates of the point in question. The CRT coordinates are 16-bit, 2's complement binary integers. The X coordinate is stored at CRTX and CRTX+1, while the Y coordinate is stored at CRTY and CRTY+1.

d. Figure 3-30 illustrates the two coordinate systems and the transformation equations involved.

3.6.20 Error Handling Routines:

a. When an error condition occurs in the navigation loop, the navigation process is halted. Rather than halt the PILOT, a group of routines exist to handle all possible error conditions. The basic philosophy is to display an error message, reset the stack pointer, and continue execution at EXLOOP. The error message informs the operator of the cause of the problem. The stack pointer must be reset so that the stack does not overflow due to a continuous error condition. EXLOOP is the beginning of the navigation loop. By continuing at EXLOOP, PILOT will usually recover from the error condition.

b. An error display field has been established on a normal PILOT navigation display below the alphanumeric field. ERC is called at the end of the navigation loop to clear the error display field.



XRW = [FLT(CRTX)•COS(Θ) − FLT (CRTY)•SIN(Θ)] /SCL + X_o

YRW = [FLT(CRTX)•SIN(Θ) + FLT (CRTY)•COS(Θ)] /SCL + Y_o

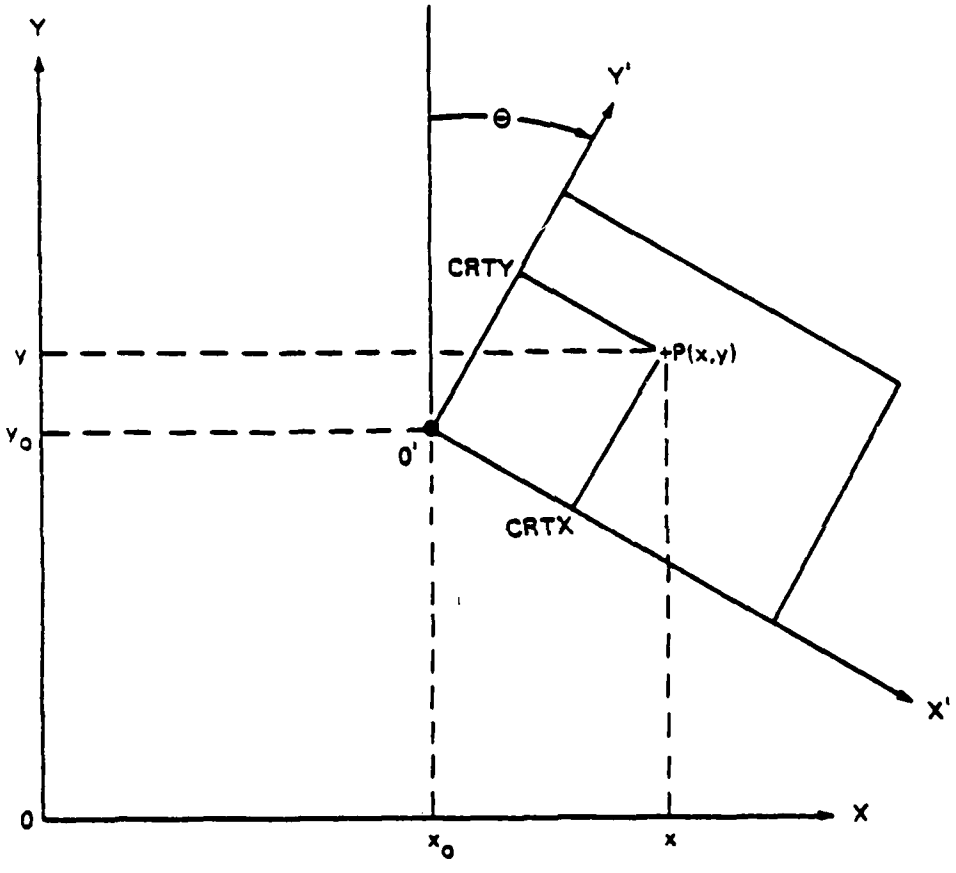
, WHERE:

SCL = SCALE FACTOR, CRT LINES/YARD

X, Y: REAL WORLD (LORAN CHAIN) COORDINATE SYSTEM

X', Y': DISPLAY COORDINATE SYSTEM

FIGURE 3-29 CRTRW COORDINATE SYSTEMS.



CRTX = FIXS $[(x-x_o)\cdot COS(\Theta) - (y-y_o)\cdot SIN(\Theta)]\cdot SCL$

CRTY = FIXS $[(y-y_o)\cdot COS(\Theta) + (x-x_o)\cdot SIN(\Theta)]\cdot SCL$

, WHERE:

SCL = SCALE FACTOR, CRT LINES/YARD

X, Y: REAL WORLD (LORAN CHAIN) COORDINATE SYSTEM

X', Y': DISPLAY COORDINATE SYSTEM

FIGURE 3-30 RWCRT COORDINATE SYSTEMS.

3.6.20.1 Routine TER: Tape read errors are handled by TER. When TER is called, the message 'TAPE READ ERROR' is displayed, the stack pointer is reset, and execution continues at EXLOOP. By continuing execution at EXLOOP the PILOT will attempt to recover from a read error, rather than halting in error loop.

3.6.20.2 Routine TDER: When TD to XY conversion routines fail to converge, TDER is called. TDER displays the message 'FAIL TO CONVERT', resets the stack pointer, and continues execution at EXLOOP.

3.6.20.3 Routine TDDER: Where two or more TD's are invalid, according to the edit process, TDDER is called. TDDER turns on the correct inverse video fields, displays the message 'NAVIGATION HALT', resets the stack pointers, and continues execution at EXLOOP. The TD status is in the accumulator when TDDER is called.

3.6.20.4 Routine TDAER: When the TD bias contains invalid entries, TDAER is called. TDAER displays the message 'BAD TD BIAS DATA', resets the stack pointer, and continues execution at EXLOOP.

3.6.20.5 Routine MPER: When the math pack status indicates an error has occurred, MPER is called. MPER extracts the return address of the routine which called MAP, displays the message 'MATH ERROR RA=' along with the return address, resets the stack pointer, and continues execution at EXLOOP. MPER uses HEXASC to convert the return address to ASCII for display.

3.6.20.6 Routine BER: During conversion from BCD to binary, if any invalid data are encountered, BER is called. BER retrieves the return address of the routine which called BCDBIN, displays the message 'BAD BCD DATA RA=' along with the return address, resets the stack pointer, and continues execution at EXLOOP. BER uses HEXASC to convert the return address to ASCII for display.

3.6.20.7 Routine ERC: The error display field is cleared when ERC is called. A call to ERC occurs at the end of the navigation loop in order to clear the error display field.

3.6.20.8 Routine HEXASC: HEXASC performs conversion of hexadecimal to ASCII. On entry DE points to the destination of the ASCII, and HL points to the byte to be converted.

3.6.21 Hewlett-Packard Software Referenced by PILOT - HPSD: Routine HPSD defines the entry vectors and variables contained in the Hewlett-Packard code which the PILOT code uses. No calls are made to HPSD; only the quantities defined herein are used to satisfy external references in PILOT.

3.6.22 Diagnostic Data: When the PILOT unit is placed in the diagnostic mode the only raw data needed are contained in subroutines DIAD or DIAD1. DIAD contains the TD's in math pack and raw data forms required to place the vessel at the waypoint. No calls are made to DIAD. DIAD is used only to obtain TD's for the diagnostic mode. DIAD1 contains the TD's in math pack and raw data forms to place the vessel at a point 300 feet from the waypoint. No calls are made to DIAD1. DIAD1 is used only to obtain TD's for the diagnostic mode.

4.0 8085 SOFTWARE RECEIVER INTERFACE.

4.1 Receiver Interface Software Package:

a. The functions of the receiver interface card are to read TD's from at most two Loran-C receivers, select the TD's for navigation, filter the navigation TD's, and provide the Hewlett-Packard 8080 microprocessor with the current navigation TD's on request. The receiver interface software has been modularized using structured programming techniques. This procedure has enabled the receiver interface software to interface any two Loran-C receivers to PILOT by including in the receiver interface software package the appropriate modules whose function is to handle the raw data from the corresponding Loran-C receiver.

b. The PILOT program is interfaced to a Teledyne 708 and Internav Mark III Loran-C receiver combination and to a Teledyne 708 and Internav LC404 Loran-C receiver combination. Two distinct hardware and software packages are utilized in these interfaces. The memory maps are basically the same for both designs (see Fig. 4-1). The only minor difference is that a USART was added at 3600 H to communicate with the Internav LC404.

c. The software package which handles the interface to the Teledyne 708 and Internav Mark III Loran-C receiver combination is controlled by an executive program, REXEC original. This version of REXEC performs all the control and decision processes required in interfacing PILOT to this receiver combination. Associated with this version of REXEC is RICOM Version 4, which defines all variables used in the receiver interface software.

d. When another receiver, the Internav LC404, was interfaced to PILOT with the Teledyne 708, the software package for the LC404 replaced the Internav Mark III software package in the existing receiver interface software. Also, certain modules had to be modified to accommodate this receiver. REXEC, RICOM, and TI were modified to Revison A level to accommodate this receiver.

e. When the 8080 microprocessor wants the current navigation TD's, the receiver interface software services an interrupt which handles this communication. Communication between the 8080 and 8035 microprocessors is via I/O ports rather than memory locations as viewed from the 8085 microprocessor.

f. The Teledyne 708 and Internav Mark III receivers output their TD's at a regular rate. The Internav LC404 receiver will output TD's only on request. This request for data from the receiver interface occurs every 0.5 second. The feature has been added to the REXEC wait-loop.

4.2 Interrupts:

4.2.1 Internav Mark III Receiver Software: The software module IRI communicates with the Internav Mark III receiver. IRI is called when an RST 6.5 interrupt occurs. IRI makes the determination if any TD data are on the data lines and if the receiver interface is in the initialization or filtering mode and then places the data in the appropriate data buffer. While servicing this interrupt all other interrupts are disabled.

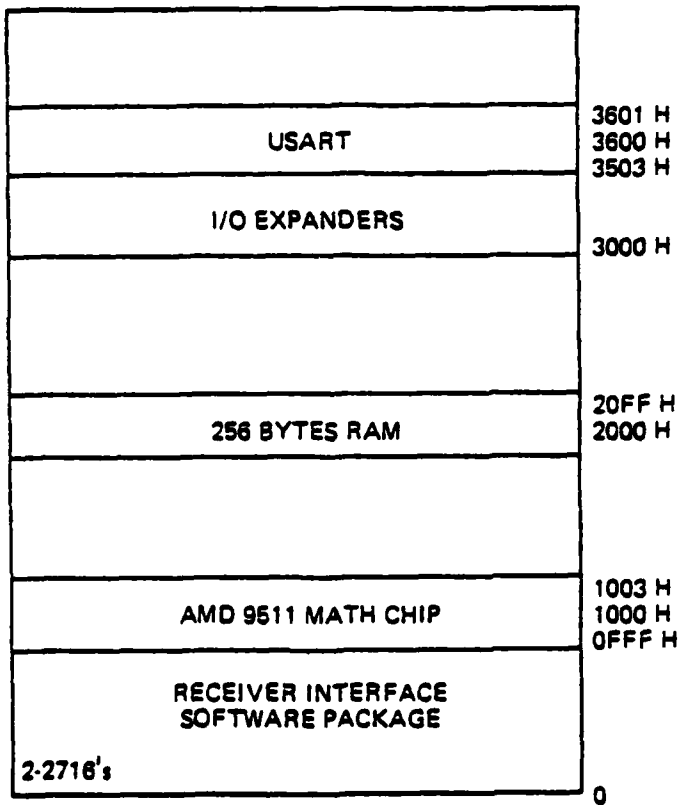


FIGURE 4-1. RECEIVER INTERFACE MEMORY MAP.

4.2.2 Internav LC404 Receiver Software Package:

a. The software to process data from the Internav LC404 receiver was written to mesh with the existing software in the receiver interface card. The LC404 software package takes the place of the Internav Mark III software package.

b. The LC404 software package consists of nine subroutines, each of which perform a part of the total interface to the LC404. Descriptions of these subroutines are in the following sections.

4.2.2.1 Subroutine FOFSU: FOFSU initializes the USART which communicates with the Internav LC404 receiver. The proper command bytes are sent to the USART to configure it for communications with a USART in the LC404 receiver.

4.2.2.2 Subroutine FDR: FDR initiates a request for TD data from the Internav LC404 receiver. See Fig. 4-2 for the flow chart of detailed operations.

4.2.2.3 Subroutine DELAYT: DELAYT generates a wait state in the processor by executing a loop N times. Where N is specified in the DE register pair, N times the amount of time required to execute the loop yields the total delay time. The amount of time required to execute the loop is a function of the system clock, which is 10 microseconds in DELAYT.

4.2.2.4 Subroutine FOFI: FOFI is the interrupt handler for the Internav LC404 receiver which is an RST 6.5. FOFI saves the registers, reads the USART states, and determines if the USART is transmitting or receiving. If USART is transmitting, FOFI jumps to FOFT. If the USART is receiving, FOFI jumps to FOFR.

4.2.2.5 Subroutine FOFT: FOFT completes the data request and, upon completion of the request message, reconfigures the USART to receive data. The basic function of FOFT is to transmit another byte to the Internav LC404 receiver. Before transmitting a check is made to determine if more data are to be transmitted. If no data are to be transmitted the USART is set to receive characters from the LC404 receiver.

4.2.2.6 Subroutine FOFR: FOFR receives characters from the Internav LC404 receiver. It also determines when all characters have been received from the receiver, sets the receiver status byte to indicate this event, stores the data from the receiver, and denotes the completion of the data from the LC404 receiver.

4.2.2.7 Subroutine FOFUP: FOFUP unpacks the raw LC404 receiver data on a TD basis and places the TD in memory. The calling routine points to the TD in the HL register pair and specifies where to put the data in the DE register pair.

4.2.2.8 Subroutine FOFID: FOFID uses FOFUP to format the LC404 data for the identification process and checks the status byte from the Internav LC404 receiver to determine if the TD's are valid according to the LC404. If the

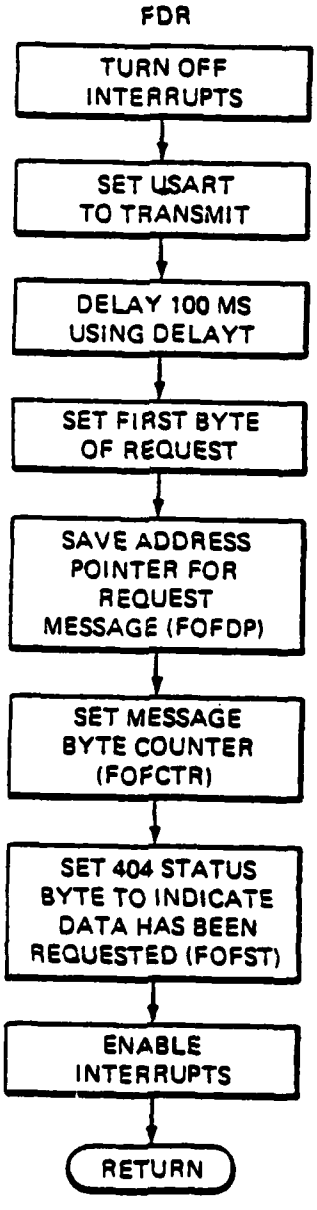


FIG. 4-2 FDR FLOW CHART.

TD's are not valid, subroutine FDR is used to request TD's again. If the TD's are valid, subroutine FOFUP is used to put the TD's in the format required to identify the TD's using IDENT.

4.2.2.9 Subroutine FOFD: FOFD uses FOFUP to format the data for use by the filters if any TD's from the LC404 receiver are being used for navigation. If any of the TD's are being used, they are unpacked via FOFUP and placed in memory as required by the filter routines.

4.2.3 Teledyne Receiver Software TRI: The TRI software module handles the Teledyne receiver raw data. When an RST 5.5 interrupt occurs, TRI is called. TRI determines if the receiver interface is in the initialization or filtering mode and reads the raw data into the appropriate buffer. When this interrupt is being serviced, all other interrupts are disabled.

4.2.4 Timer Interrupt in the Receiver Interface:

a. The timer interrupt in the receiver interface serves as a real-time clock with 100-millisecond tick marks. This 100-millisecond interval has been determined as the rate at which the TD filters will be dead reckoned (DR). If raw data are available the raw data will be incorporated into the filter after the DR process.

b. While updating the filters the receiver interface interrupts will be active to avoid losing any data from the receivers. The subroutines associated with the timer interrupt are described in the following sections.

4.2.4.1 Subroutine TIR: TIR services the timer interface interrupt which is the TRAP interrupt. Every 100 milliseconds the hardware clock issues an interrupt which is serviced by TIR. TIR sets a flag, ATI, which is examined by the wait-loop to determine if a timer interrupt has occurred and reenables the interrupts.

4.2.4.2 Subroutine TI:

a. TI maintains three independent time counters, which are used as binary clocks by the filter update routines, and updates the TD filters. TI examines each of three possible TD running filters to determine if they are running. Any active filter is updated using subroutine TDDR. After examining all TD's the buffer pointer is updated to point to the current buffer. TI then uses EXLOOP to incorporate any available raw data into the TD filters.

b. Revision A of TI performs the same functions as TI. An extra counter, DPC, has been added to detect when 0.5 second intervals for interrogating the Internav LC404 receiver occur.

4.2.4.3 Subroutine TDDR:

a. TDDR performs only the DR process. If raw data are available EXLOOP incorporates it into the TD filters. Associated with each TD is a TD and a TD. When the TD is updated, TD is also updated.

b. TDDR performs the following calculations:

$$TD_i = TD_{i-1} + \Delta t \ \dot{TD}_{i-1} + \frac{\Delta t^2 \ \ddot{TD}}{2}$$

$$\dot{TD}_i = \dot{TD}_{i-1} + \Delta t \ \ddot{TD}$$

To accommodate all modes of operation of these equations it is necessary to maintain a double-precision floating-point part and a double-precision fixed-point part of each TD.

4.2.4.4 Subroutine FILTER: FILTER incorporates a new TD value into the current TD filter value. FILTER executes the following equations:

$$TD = TD_i + \left(\frac{A + B * DT_A}{\Delta t}\right) \left(TD_M + DT_A * \dot{TD}_i + DT_A^2 * \ddot{TD} - TD_i\right)$$

$$\dot{TD} = \dot{TD}_i + \frac{B}{\Delta t} \left(TD_M + DT_A * \dot{TD}_i + \frac{DT_A^2 \ \ddot{TD}}{2} - TD_i\right).$$

4.2.5 Hewlett-Packard Interrupt Handling in the 8085:

a. The Hewlett-Packard terminal can interrupt the 8085 using the RST 7.5 of the 8085. When this occurs, the 8085 jumps to HPIH, Hewlett-Packard interrupt handler. HPIH reads the command byte which is in Port A and the status in Port C. The command byte is decoded to determine what data exchange is to follow and calls one of the following routines: TDIO, TDIDEN, TDFC, MEMD, DIAG or DIAG1.

b. Each routine handles a different data exchange between the 8080 and 8085 in the 8085. Descriptions of the functions of these routines are contained in the following sections:

4.2.5.1 Routine TDIO: TDIO is called by HPIH to exchange status, TD's, TD's, and TD's between, the 8080 and 8085. It uses STACW and STACR to determine when data are to be read from or placed in the appropriate ports. Data are sent in the following order to the 8080: status, TD's, TD's, The TD's are then sent from the 8080.

4.2.5.2 Routine TDIDEN: TDIDEN is called by HPID to receive identifying TD's from the 8080. It uses STACR to determine when the 8080 has placed data in Port B to be read. Three sets of TD's are sent from the 8080 and are to be used to select the desirable navigation TD's from the receivers.

4.2.5.3 Routine TDFC: TDFC is called by HPIH to receive filter constants from the 8080, and it uses STACR to determine when the 8080 has placed data in Port B to be read. Six sets of filter constants are sent from the 8080, and

after placing these constants in RAM a check is made to determine if this is the initial start up. If initial start up is determined, a flag is cleared to indicate the filter constants have been received. If there is no initial start, the TD filters are reinitialized using the current TD's from the receivers.

4.2.5.4 Routine MEMD: MEMD is called by HPID to send the entire 8085 RAM from 2040 H to 20FF H to the 8080. This is used only for checkout purposes; this routine is never used during normal operation.

4.2.5.5 Routine DIAG: DIAG is called by HPIH to place the 8085 in a diagnostic mode. In the diagnostic mode, the receiver interrupts are disabled, and TD's are placed in the input buffer as though the receiver had sent them. The 8085 then operates as usual, assuming the TD's were from the receivers. The TD's place the vessel at the waypoint.

4.2.5.6 Routine DIAG1: DIAG1 performs the same as DIAG, except that the TD's place the vessel at a point 300 feet from the waypoint.

4.2.5.7 STACW - Check Status Port B: STACW checks the status port, address 23 H of the 8085, to determine when Port B data have been read by the 8080. When STACW is called in normal use, it continually checks the status until bit 4 is set. Control returns to the calling routine when this occurs. If this bit is not set within 65,000 checks, i.e., FFFF H, then control passes to the error routine, EREX. No parameters are passed.

4.2.5.8 STACR - Checks Status Port A: STACR checks the status port, address 23 H of the 8085, to determine when Port A has been loaded with data by the 8080. When STACR is called in normal use, it continually checks the status port until the two least significant bits are set. Control returns to the calling routine when this occurs. If these bits are not set within 65,000 checks, i.e., FFFF H, then control passes to the error routine, EREX. No parameters are passed.

4.3 EXECUTIVE.

REXEC consists of two main sections, the receiver interface control and TD filter update software. The control section performs the setup and initialization required to place the processor in the TD filtering mode. After having performed these functions REXEC is permanently in an inactive wait-loop. At this time the filtering process is interrupt driven. When receivers have data available, the data are read into raw data buffers. At a 0.1 second interval, the TD filters are dead-reckoned (DR) to a new value. The TD filter update section then decides if the raw data buffers contain new data. If they contain new data, then each TD filter is updated by a new value using the routine FILTER.

4.3.1 Routine REXEC: REXEC is the executive routine for the receiver interface software package. REXEC is not called by any routine and performs all the control and decisions in the receiver interface. REXEC interfaces the Teledyne 708 and Internav Mark III receiver to PILOT. Another version of REXEC performs identically, except that this software interfaces a Teledyne 708 and Internav LC404 receiver to PILOT.

4.3.2 Routine RICOM: RICOM defines all variables used in the receiver interface software. The original version of RICOM is used with the original version of REXEC. No calls are made to RICOM, only the variables defined herein are used. Revision A of RICOM is used with revision A of REXEC.

4.3.3 TD Identification Process: Before the PILOT unit can begin navigation, the receiver interface software must determine the origin of the TD's. The receiver interface could be connected to two receivers, each of which outputs four TD's for a total of eight TD's. Routines IDENT and TDIM perform the functions on the TD's which are described in the following sections.

4.3.3.1 Routine IDENT: The TD identification process is performed by IDENT. By using the value of a maximum of three TD's, IDENT determines if any of the TD's from the receivers are within 1000 microseconds of any of these three TD's. When a TD from a receiver is identified, an identification byte is set to signify that the TD will be used for navigation. The receiver time delay is also set according to the source of the TD. When all TD's from both receivers are examined, the receiver interface software is then ready to enable the filters. IDENT uses TDIM to perform the mathematical operations involved in the TD identification process.

4.3.3.2 Routine TDIM: The mathematical operations involved in the TD identification process are performed by TDIM. TDIM performs the following calculation:

$$1 \text{ millisecond} - |TDX - TDI|$$

where TDX is some unknown TD and TDI is a known TD. If the result of the calculation is positive, the TDX is assumed to be TDI. The address of TDI is in the DE register pair and the address of TDX is in the HL register pair when TDIM is called. On exit the accumulator contains the math pack status which contains the sign of the result of the calculation.

4.3.4 Memory Initialization in the Receiver Interface - MEMI: MEMI initializes the RAM area of the receiver interface from the initial turn on configuration to the filter mode. When the receiver interface is powered on, the RAM area is configured to identify which TD's from which receiver will be used. The stack is cleaned up by resetting the stack pointer, and the TD buffer and alternate TD buffer scheme are established.

When this process is completed the filters are then enabled. Prior to enabling the filters, MEMI reconfigures RAM for the filter process. The status flags are set and cleared as required by data logging.

4.3.5 CHATD - Change in TD Module:

a. Routine CHATD computes the change in TD from the previous update to the present update and compares this change to a "window" value. If the change in TD falls within this window, then the new TD is considered "good"; otherwise, it is considered "bad" and is rejected. CHATD is called by the receiver interface executive following input of a TD from the receiver.

The executive first calls BCDBIN to convert the raw TD to its 32-bit binary integer equivalent and then calls CHATD to compare the present TD value with the previous one.

b. The CHATD routine operates in the following manner. Based on maximum allowable vessel dynamics and the rates-of-change of the TD's, a TD "window" value is specified. This window defines the maximum expected change in TD if the vessel is undergoing the maximum dynamics (e.g., a full-speed, hard-turn situation). CHATD subtracts the previous TD, which has been saved from the present TD to produce a change-in-TD value.

c. The absolute value of the change-in-TD value is then compared to this "window" specification. If the change-in-TD value falls within the window, the TD is "good". Otherwise, it is considered "bad" and is not used in the TD transformation algorithm. Symbolically, if W represents the window which is nanoseconds, then CHATD performs the following operation:

$$| TD.NEW - TD.OLD | \quad \leq W?$$

where the question mark (?) signifies the comparison results. The window width, W, is currently set at 3000 nanoseconds.

d. The receiver interface software sets up the TD status word, which is then sent to the Intel 8080 mainframe Central Processing Unit (CPU) and identifies the "good" and "bad" TD's.

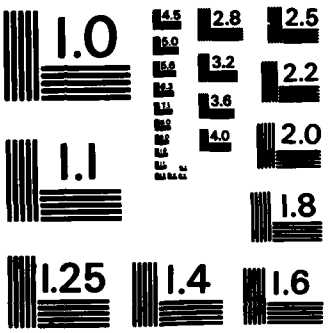
4.3.6 Error Handler in the 8085 - EREX: EREX is the error handling routine for the 8085 microprocessor and is called when an error occurs in the receiver interface software. This routine turns off all interrupts except the 8080 interrupt and sets a status byte to indicate the problem. The receiver interface will remain in EREX until the processor is reset. The software has assumed that when EREX is called, a fatal error has occurred and the only way to recover is with a hard reset.

4.3.7 BCD-to-Binary Conversion Module - BCDBIN:

a. BCDBIN converts a seven-digit number to its equivalent 32-bit binary integer value. It is called by the receiver interface executive each time a time difference must be converted.

b. BCDBIN converts the seven-digit BCD number to binary form. BCDBIN uses a direct conversion algorithm, i.e., the least significant byte (the status byte) is multiplied by zero and added to an accumulator. The next byte is multiplied by 10 and summed. The third byte is multiplied by 100 and summed. This process continues until the most significant byte is multiplied by 10^7 and summed into the 32-bit binary accumulator. The final sum of the products represents the 32-bit binary integer equivalent of the seven-digit BCD input number. The 32-bit binary integer is retrieved from the math pack and stored in RAM memory.

AD-A121 759 PILOT: A PRECISION INTERCOASTAL LORAN TRANSLOCATOR 2/2
VOLUME 3 SOFTWARE(U) JOHNS HOPKINS UNIV LAUREL MD
APPLIED PHYSICS LAB G E BAER ET AL. MAR 82
UNCLASSIFIED USCG-D-21-81-3 N00024-78-C-5384 F/G 9/2 NL



c. The powers of 10 are coded as a fixed table. Each power of 10 is represented by a 32-bit binary integer (four bytes). The greatest power of 10 is 10^7, and the least power of 10 is 10^1. The powers of 10 table has the name FACTOR. FACTOR addresses the most significant byte of the most significant power of 10 ($=10^7$).

d. BCDBIN also performs a validity check on each of the BCD digits it sends to the math pack, i.e., the BCD digits must fall in the numerical range of zero to nine. If a digit is outside of this range, an error condition exists, and BCDBIN jumps to an error processing module, EREX.

5.0 MATH HANDLERS.

5.1 Basic Math Pack Handler Routines: The AMD 9511 math chip requires that it receives data along with a command, and that after the command is executed, the result is retrieved from the math chip. The functions performed by the processors require numerous math operations. To simplify software coding, three software modules -- LDMP, MPRDY, and STMP -- have been written which communicate with the math chip. These modules are discussed in Sections 5.1.1 through 5.1.4.

5.1.1 Routine LDMP1: LDMP1 transfers one 4-byte operand and a command to the AMD 9511 math chip.

Entry Conditions:

HL contains address of LSB of operand B.
C contains command to be executed.

Exit Conditions:

HL contains MSB address of operand B.
A & C contain command.

5.1.2 Routine LDMP2: LDMP2 transfers two 4-byte operands and a command to the AMD 9511 math chip.

Entry Conditions:

HL contains address of LSB of operand A.
DE contains address of LSB of operand B.
C contains command to execute.

Exit Conditions:

HL contains MSB address of operand B.
DE contains MSB address of operand A.
A & C contain command.

5.1.3 Routine STMP: STMP will read a result from the math pack and store it at the location specified in the HL registers:

Entry Condition:

HL contains MSB address of result.

Exit Conditions:

HL contains LSB address of result.
A register destroyed.

5.1.4 Routine MPRDY: MPRDY checks to see if the math pack is ready (i.e., has completed the last operation) with no errors. When the math pack is ready with no errors, control is returned to the calling routine. If the math pack is not ready, or has an error, control is passed to EREX (see Section 4.3.6) which halts the processor.

Entry Conditions: None

Exit Conditions: No registers destroyed.

5.1.5 Math Pack Definitions - MPDEF:

a. MPDEF is a data module which defines symbols and mnemonics associated with AMD 9511 arithmetic processing unit (APU). It is a non-executable module containing only data and definitions. The PILOT system contains two APU's: the APU-85, used by the receiver interface 8085 processor, and the APU-80, used by the mainframe 8080 processor. The APU-80, also mounted on the receiver interface card, is treated as memory-mapped I/O by the 8080 processor. From an 8080 standpoint, the APU-80 resides at four addresses: 8A00H through 8A03H. These addresses have been named according to their functions:

```
MPIN  EQU 8A00H -- Data Input Port
MPCMD EQU 8A01H -- Command Input Port
MPOUT EQU 8A02H -- Data Output Port
MPSAT EQU 8A03H -- Status Output Port
```

b. The receiver interface hardware decodes these addresses and sets up and executes the basic software math handlers (LDMP, MPRDY, and STMP) used to communicate between the 8080 CPU and APU-80. Communication is across the "bottom-plane bus." Hardware operation and details can be found in JHU/APL SDO 5699.1, PILOT, A Precision Intercoastal Loran Translocator, Volume II - Hardware, July 1980.

5.1.6 Math Pack Routines in the Receiver Interface: The receiver interfaces uses the same math pack routines (LDMP, MPRDY, and STMP) described in Sections 5.1.1 through 5.1.4. The locations of the math pack and the math pack instructions are defined in MPDEF5 for the receiver interface. No calls are made to MPDEF5 only the variables defined herein are used by other programs in the receiver interface.

5.1.7 Zero the Math Pack - ZRMP: ZRMP is a utility module that will clear the math pack stack. The stack for the AMD 9511 math processing chip consists of a 16-byte linear array. The array operates as a push-down stack and can be treated as four, 32-bit operands for double-precision and floating point operations, or as eight, 16-bit operands for single-precision operations. When called, ZRMP will set to zero all 16 bytes of the math pack stack.

5.2 Revised Math Routine - MAP:

a. To reduce the coding requirements associated with the math pack handlers described in Section 5.1, the routine MAP was developed. MAP is used primarily in the transformation software.

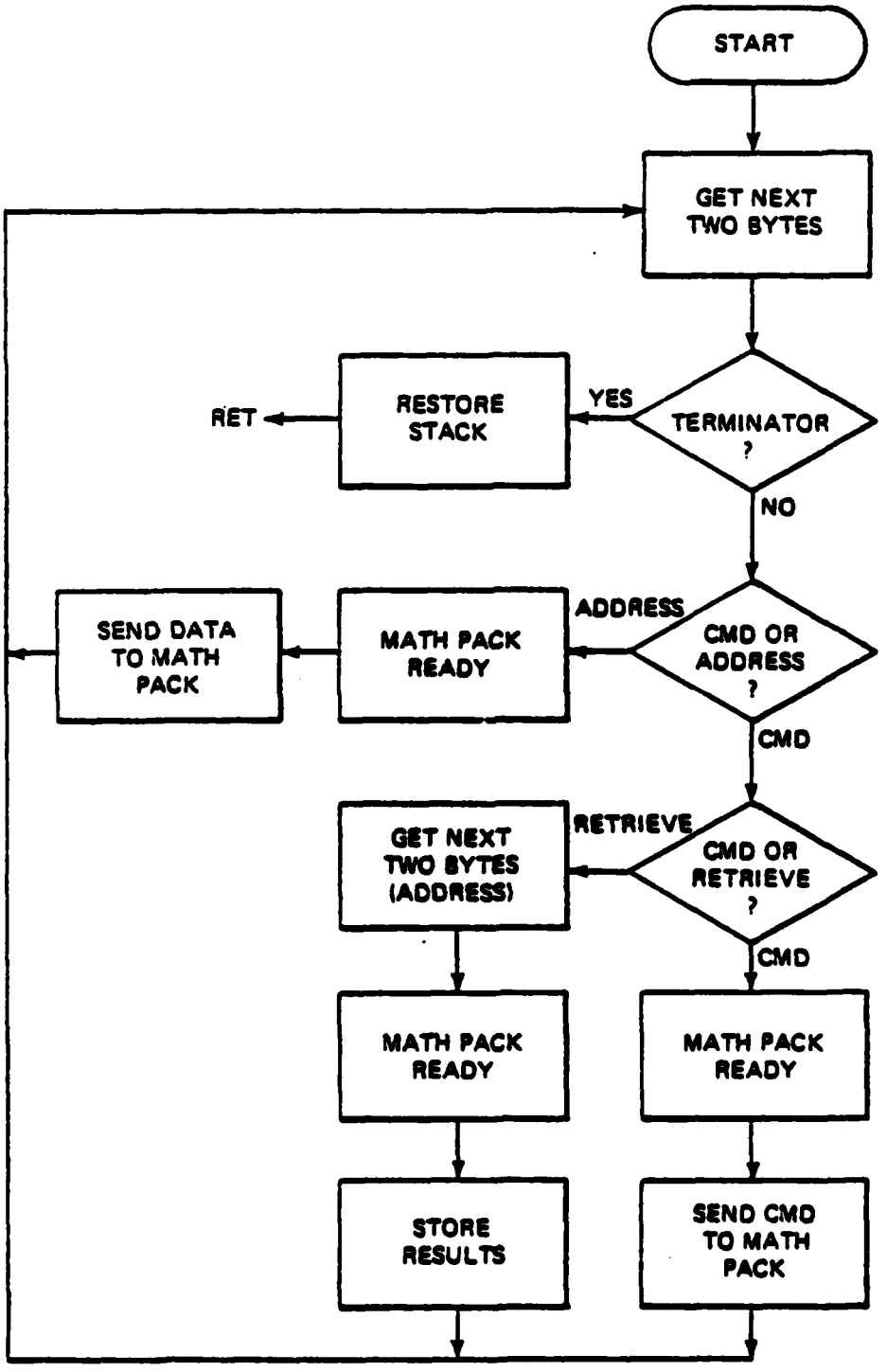


FIGURE 5-1 ROUTINE MAP.

b. The flow chart used for MAP is shown in Fig. 5-1. To use MAP the addresses of the data and the actual operands follow the call to the respective program. The handler software decodes the data and operands to take the appropriate action.

c. All operations are defined in MAPDEF. These operations are the standard operations defined in the AMD 9511A data sheet (Appendix F) with the exception of operations MPNOP, RETD, and TERM. MPNOP is the math pack NOP. NOP is already a microprocessor mnemonic and, therefore, could not be used with respect to the math chip. RETD signals MAP to retrieve a word of data from the math chip and place that data at the address specified following the RETD command. TERM informs MAP to terminate math pack commands and return to the calling program.

d. The address of any operand always refers to the Most Significant Byte (MSB) of the operand. MAP adjusts the address of the operand as needed to point to the LSB or MSB. The handler software requires the address of the operand to point to the MSB and the address of the operand plus three to point to the LSB.

5.3 Math Pack Routines in the Receiver Interface: The receiver interface uses the same math pack routines (LDMP, MPRDY, and STMP) described in Sections 5.1.1 through 5.1.4. The locations of the math pack and the math pack instructions are defined in MPDEF5 for the receiver interface. No calls are made to MPDEF5, only the variables defined herein are used by other programs in the receiver interface. The APU-85, mounted on the receiver interface card, is treated as memory-mapped I/O by the 8085 processor. From an 8085 standpoint, the APU-85 resides at four addresses: 1000 H through 1003 H. These addresses have been named according to their functions:

```
MPIN   EQU 1000 H -- Data Input Port
MPCMD  EQU 1001 H -- Command Input Port
MPOUT  EQU 1002 H -- Data Output Port
MPSTAT EQU 1003 H -- Status Output Port
```

6.0 TAPE CONVERT.

6.1 Tape Convert Overview:

a. The Tape Convert software is a stand-alone module used to convert navigation source tapes to object tapes which are used by the PILOT terminal. Tape Convert has four modes of operation which are selected by the programmable function keys f1, f4, f5, and f8. The four keys are labeled:

f1 - START
f4 - DISPLAY FILE
f5 - CONVERT FILE
f8 - CONVERT 7 FILES

b. A terminal reset (hard or soft) will load the four function keys (f1, f4, f5, and f8) with the proper escape sequences for Tape Conversion operation. This relieves the operator from having to define the function keys each time the Tape Convert program is to be used.

c. A description of each of the four Tape Convert function keys follows:

1. f1 - START. The START mode rewinds both tapes and copies files 1 and 2 from the source tape (left tape unit) to the object tape (right tape unit). It is used to copy the index file (file 1) and the reserved file (file 2) directly from the source tape to the object tape. The START key functions as follows:

   (a) Find file 1 on left tape.
   (b) Find file 1 on right tape.
   (c) Copy 2 files from left to right tape.
   (d) Home the cursor and clear the screen.
   (e) Display the following message on the screen to indicate the completion of the START mode.

   'FILES 1 and 2 COPIED'

2. f4 - DISPLAY FILE. The DISPLAY file mode reads one file from the source tape (left tape unit) to the screen in the display functions mode. It is used to display a source file to which changes need to be made. The DISPLAY FILE key functions as follows:

   (a) Home the cursor and clear the screen.
   (b) Turn display functions mode on.
   (c) Read a file from the left tape to the screen.
   (d) Turn display functions mode off.
   (e) Home the cursor.

3. f5 - CONVERT FILE.

(a) The CONVERT FILE mode converts the source file that is displayed on the screen. It is used to convert a source file to which changes have been made by the operator. The CONVERT FILE key functions as follows:

(1) Home the cursor.
(2) Read a file from the screen to memory.
(3) Perform the tape conversion.
(4) Write the object file to the right tape.
(5) Display the following message on the screen to indicate the completion of the CONVERT FILE mode:

'***TAPE CONVERT COMPLETED***'

(b) The Tape Convert software looks at row 1, column 16 of the source file to decide whether a master file or a detail file is being converted. An "M" in this location indicates a master file, and a "D" indicates a detail file. If any other character appears in this location, then an error message is displayed on the screen and no file conversion occurs. The error message displayed to the operator is:

'***ERROR - FILE NOT MASTER OR DETAIL***'

4. f8 - CONVERT ╖ FILES.

(a) The CONVERT ╖ FILES mode converts files directly from the source tape (left tape unit) to the object tape (right tape unit). The operator selects the number of files to be converted. It is used to convert source files to which no corrections need to be made. The CONVERT ╖ FILES key functions as follows:

(1) Home the cursor and clear the screen.
(2) Display the following message on the screen:

'CONVERT ╖ FILES'

The character ╖ is displayed inside a two-column wide inverse video field.)
(3) The operator must then enter a one- or two-digit number (range 1-99), followed by a carriage return, to select the number of files to convert. (The entered characters are echoed back on the screen inside the inverse video field.)
(4) Read a file from the left tape into memory.
(5) Perform the tape conversion.
(6) Write the object file to the right tape.

(7) Repeat steps 4-6 above until the correct number of files have been converted, or until the end of data mark is detected after reading to the left tape.

(8) Display the following message on the screen to indicate the completion of the CONVERT ⇃ FILES mode:

'***TAPE CONVERT COMPLETED***'

(b) The operator must enter a one- or two-digit number from the keyboard in step 3 above. Only numerical characters (0-9) can be entered. If any other key is hit, it is ignored, and the bell will sound. If a third numerical character is entered, an error message is immediately displayed, and the CONVERT ⇃ FILES mode is aborted. The error message reads:

'***ILLEGAL KEYBOARD INPUT - TAPE CONVERT ABORTED***'

(c) The number entered cannot be zero. If a zero character followed by a carriage return (or only a carriage return) is entered, then the message above is displayed, and this mode is aborted.

(d) Because only the numerical keys are active, the operator cannot change an entered number by back-spacing the cursor. The only option to the operator would be to enter a three-digit number, which aborts this mode. The operator can then select the CONVERT ⇃ FILES mode again to enter the correct number.

(e) The operator can enter a number greater than the number of files on the source tape. If an end of data mark is detected after reading the source tape, then the tape conversion is ended.

(f) After the one- or two-digit number is entered, the operator must hit the carriage return key to start the Tape Convert process. The inverse video field around the entered number is removed, and the tape conversion begins. A message is displayed to the operator after all files have been converted.

(g) The Tape Convert software looks at row 1, column 16 of the source file to decide whether a master file or a detail file is being converted, as described in the CONVERT FILE mode. If any source file contains an illegal character at this location, then an error message is displayed, and the tape conversion stops at this point.

(h) If an error is detected during a tape read operation, or if an error condition occurs during a math pack operation, then an error message is displayed, and the terminal halts. The error message reads:

'FATAL ERROR - RESTART REQUIRED'

The most likely cause of a math pack error is a wrong format number or a number out of bound on the source file. A terminal resest is required if this error occurs.

6.2 Tape Convert Software - TC:

a. The Tape Convert (TC) software resides in the HP terminal along with the HP operational software. Normal terminal operation is not affected by the addition of the Tape Convert software. Tape Convert resides in 4K bytes of memory, beginning at location C000 H. The TC software is contained in two EPROM's, at location C000 H and C800 H, and extends to location C958 H, leaving adequate space for expansion in the second EPROM. The TC software uses RAM from location D600 H to DFFF H. A memory map for the TC software is shown in Fig. 6-1.

b. The link procedure requires both the TC disk and the UTIL disk.

c. To incorporate the TC software into the terminal, some patches were made to the HP software module HP00. These changes are shown in Fig. 6-2. The patches were made to change some memory bounds and to program the function keys for TC operation. In addition to the two TC EPROM's at locations C000 H and C800 H, an EPROM for module HP00 (located at 0000 H), which includes the above patches, must also be installed in the terminal.

d. Entry is made into the TC software via four programmable function keys (f1, f4, f5, and f8). The function keys are loaded with the proper escape sequence when the RESET TERMINAL key is hit.

e. The first four instructions in executive program TCONV are the entry vectors into the TC software. These locations are used in the escape sequences that program the function keys. The first three entry vectors are used for three of the function keys, the fourth entry vector is used for the RESET key. One function key executes entirely by escape sequence.

f. Figure 6-3 shows a flow diagram for the TC executive program. Entry into the executive is via the four programmable function keys (f1, f4, f5, and f8) or the RESET key. Exit from the executive is always made into the HP wait-loop (location 294 H).

g. Function key f1 (START mode) executes entirely from escape sequence. It copies files 1 and 2 from the left tape (source) to the right tape (object).

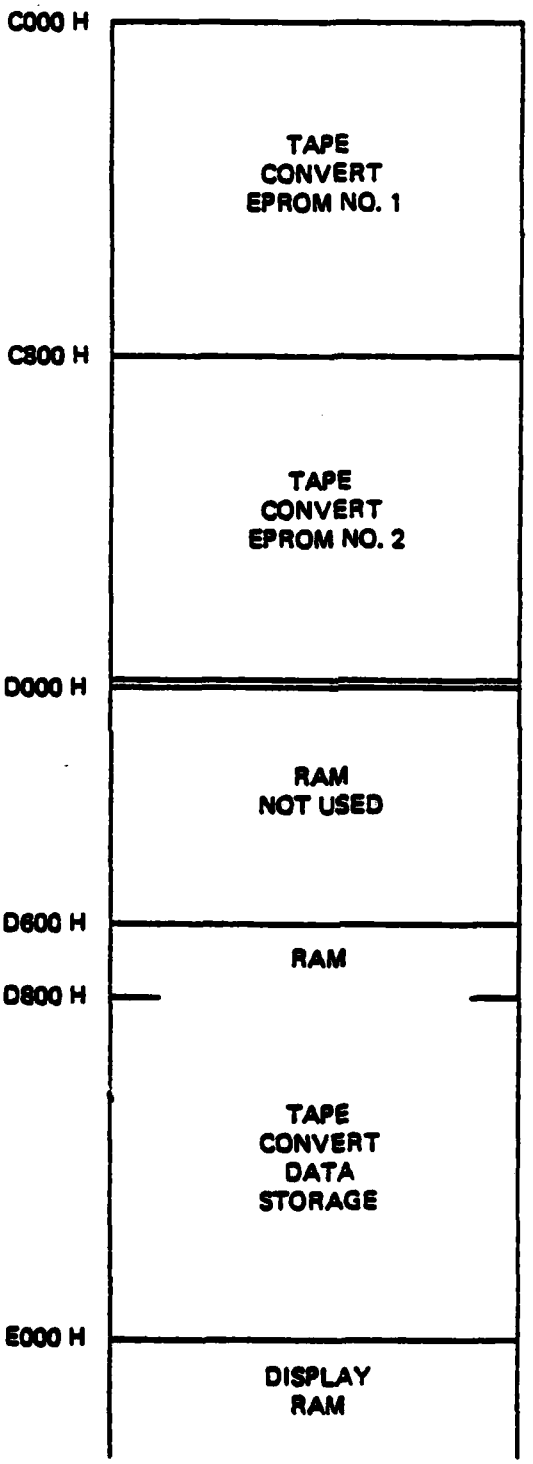


FIGURE 6-1 TAPE CONVERT MEMORY MAP.

LOCATION	WAS	CHANGE TO	COMMENT
156 H	CFH	DFH	RAM TEST UPPER BOUND
15BH	BOH	DOH	RAM TEST LOWER BOUND
169 H	DOH	EOH	DISPLAY MEMORY LOWER BOUND
291 H	32H	C3H	JMP RESET TO
292 H	6CH	09H	PROGRAM FUNCTION KEYS
293 H	FFH	COH	

FIGURE 6-2 HP00 MODULE.

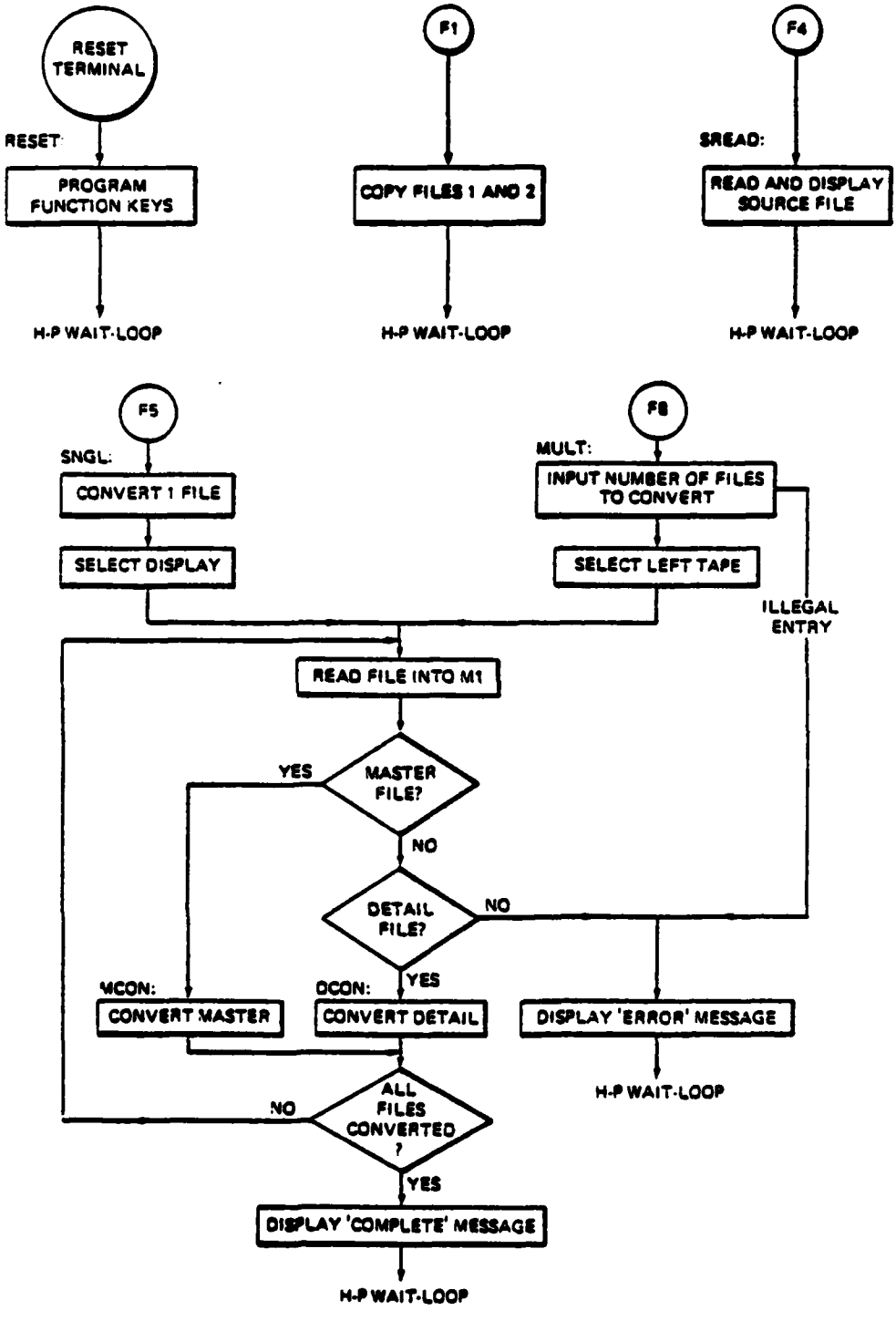


FIGURE 6-3 TAPE CONVERT EXECUTIVE.

h. Function key f4 (DISPLAY FILE mode) is programmed to jump to subroutine SREAD, which reads a source file from the left tape to the screen in the display functions mode.

i. Function key f5 (CONVERT FILE mode) is programmed to jump into TCONV at entry point SNGL. This mode converts the source file that is displayed on the screen. Before entering the main executive loop, the screen is selected as the source file input device, and the number of files to convert is set to one.

j. Function key f8 (CONVERT n FILES mode) will convert an operator-selected number of files directly from the source tape. This key is programmed to jump into TCONV at entry point MULT. Before entering the main executive loop, the number of files to convert is entered by the operator. Subroutine INPUT controls and processes the keyboard entries. The left tape is selected as the source file input device.

k. Once in the main executive loop, a source file is read into memory (at location M1) from the selected input device. To determine whether master or detail processing should follow, a check is made on the character at location M1-15 bytes. If it is an ASCII "M," master file conversion is selected; if it is an ASCII "D," detail file conversion follows. Any other character at this location will terminate the Tape Convert mode. This executive loop will continue until the selected number of files has been converted. Messages are displayed to indicate when an error condition occurs, or when processing is completed.

6.2.1 Tape Convert Routines:

a. Routine TCONV is the Tape Convert executive program. It contains the entry vectors for the function keys and the terminal reset key. TCONV includes the entry points to convert one file from the screen (SNGL), and to convert multiple files from the source tape (MULT). The source file is read into memory from the selected device. The executive selects master or detail processing on the input file. The loop continues until all files are converted. Upon completion, a message is output to the display, and TCONV returns to the HP wait-loop.

b. Routine SREAD reads a source file from the left tape unit and displays it on a screen in DISPLAY FUNCTIONS mode. The DISPLAY FUNCTIONS mode is turned off upon exit. SREAD is entered by the operator selection of the appropriate function key. The entry vector to SREAD is contained in TCONV. SREAD returns to the HP wait-loop.

c. Routine RESET programs the soft function keys for the Tape Convert entry points. RESET is called by HP software module HP00 at location 291 H. It will be called when the terminal reset key is hit. The escape sequence that loads the function key definitions is located at PFK in module TCMSG. RESET returns to the HP wait-loop.

d. Routine TCMSG contains the operator messages and the escape sequences that are used in the Tape Convert software. TCMSG is a non-executable module.

e. Routine INPUT processes a two-digit number entered via the keyboard by the operator. INPUT waits for characters to be entered. Non-numerical keys are not accepted, and the bell will sound if they are hit. The carriage return key indicates the end of operator input. An illegal input is no characters, a zero, or more than two digits. INPUT is called by TCONV.

f. Routine RAF reads a file from the selected input device and stores the data in memory. RAF stores a 00 H byte in memory following the end of data. RAF will exit on a tape end of file, tape end of data, or a display end of memory. On a tape end of data, RAF sets variable NFILE = -1. RAF is called by TCONV.

g. Routines DMSG/DCHAR add messages onto the screen. DMSG will display the message at the line under the present cursor position starting at column 4. DCHAR will display the message starting at the present cursor position. The ASCII message must be terminated with a 00 H byte.

h. Routines ESCI/ESCS execute an escape sequence. The escape sequence issues a command to the HP terminal. The escape sequence begins with an escape character (1B H) and terminates with a null character (00 H). Entry point ESCS allows an ASCII string to be output to the terminal display.

6.2.2 Master/Detail File Conversion:

a. The Tape Convert software contains two subroutines, MCON and DCON, which convert a source file to an object tape file. Subroutine MCON converts a master file, and subroutine DCON converts a detail file. The Tape Convert executive, TCONV, calls the appropriate subroutine after a source file has been read into memory.

b. Subroutine MCON converts a master source file to object tape format. The ASCII data from the source file must appear in memory beginning at location M1 (D797 H). The master source file consists of 12 lines of alphanumeric data, followed by up to 15 lines of graphics data. There is space in memory for 1232 bytes of master graphics data. The alphanumeric data appear in specifically defined fields. The beginning of each field is defined in the TCOM module under the Master File - Screen Format section. The source data in memory includes some ASCII control characters in addition to the numerical characters. Each line in the source file begins with a D3 control character, and each line ends with a carriage return and a line feed character.

c. ASCII conversion subroutines ASCFP and ASCSF require that the ASCII string not cross a 256-byte boundary in memory. This occurs three times in the master file: at location M248+1 (D800 H), M63+5 (D900 H), and M948 (DA00 H). Therefore, the data field beginning at location M248 is restricted to the last 14 characters, and the field beginning at location M63 is restricted to the last 10 characters.

d. Double fixed-point data (located in fields M53 and M73) must appear right-justified in the 15-character field. Subroutine ASCDF is called to convert an eight-character string which begins at locations M53+7 or M73+7.

e. After converting the ASCII data from the source file into the appropriate floating or fixed-point format, MCON will write the object file to the right tape unit. The first record in the master object file contains the converted alphanumeric data. The record size is 253 bytes. The alphanumeric record is followed by up to five graphics records. Each graphic record contains three lines of graphics data. The graphics data are transferred from the source data that appear in memory to the object tape. An end-of-file mark is written to the object tape following the last graphics record.

f. Subroutine DCON converts a detail source file to object file format. The source data must appear in memory beginning at location D1 (0797 H). The detail source file consists of five lines of alphanumeric data followed by up to 10 lines of graphics data. There is space in memory for 813 bytes of graphics data.

g. The TCOM module defines the start address for each field in the alphanumeric section of the detail file. The detail file includes one 256-byte boundary, at location D248+1 (D800 H). The data in this field are restricted to the last 14 characters. There are three double fixed-point parameters in the detail file, located at D53, D518, and D533. The data in these fields must be right justified.

h. After converting the ASCII source data into floating- or fixed-point format, DCON will write the object file to the right tape unit. The first record in the detail object file consists of the converted alphanumeric data (123 bytes), followed by up to three graphics records, with each record consisting of three lines of graphics data. An end-of-file mark is written to the object tape following the last detail graphics record.

i. Detailed descriptions of the various subroutines associated with file conversion are in the following sections.

6.2.3 Master File Conversion - MCON: Subroutine MCON converts a master source file to object tape format. The master file must appear in memory at location M1. The object tape records are written to the right tape unit, and an end-of-file mark is written following the object file. MCON is called by the Tape Convert executive, TCONV. TCONV is discussed in Section 6.2.1.

6.2.4 Detail File Conversion - DCON: Subroutine DCON converts a detail source file to object tape format. The detail source file must appear in memory at location D1. The object tape records are written to the right tape unit, and an end-of-file mark is written following the object file. DCON is called by the Tape Convert executive, TCONV.

6.2.5 Tape Convert Common Variables - TCOM: Subroutine TCOM contains all common variables used in the Tape Convert software. TCOM variables are contained in the memory space from D600 H to DFFF H. TCOM is a non-executable module.

6.2.6 Write a Record - WAR: Subroutine WAR writes a data record to the right tape unit from memory. After issuing the write record command, WAR monitors tape motion status and exits when tape motion stops. The maximum record length is 256 bytes.

6.2.7 Write End of File - WAEOF: Subroutine WAEOF writes an end-of-file (EOF) mark to the right tape unit. After issuing the write EOF command, WAEOF monitors tape motion status and exits when tape motion stops.

6.2.8 Graphics End of Data - GRAX: Subroutine GRAX is called before writing a graphics data record to the object tape. GRAX has three functions: (1) to look ahead through three lines of graphics data for the end of data (EOD) mark (a 00 H byte), (2) to determine the number of bytes contained in the three lines, and (3) to save the start address of the next three lines of graphics data. Subroutine GRAX1 performs the same functions, except that it looks through only one line of graphics data.

6.2.9 Space Check - SPC: Subroutine SPC searches an ASCII string for the first non-space ASCII character. The address of the first non-space ASCII character is returned to the calling program.

6.2.10 ASCII to Single Fixed Point - ASCSF: Subroutine ASCSF converts an ASCII string to a single precision, fixed-point number (2 bytes). The ASCII string must be left justified (no leading space characters). This can be performed by calling subroutine SPC, which skips over leading spaces in the ASCII string. The ASCII string must not cross over a 256-byte boundary in memory.

6.2.11 ASCII to Double Precision Fixed Point - ASCDF: Subroutine ASCDF converts an ASCII string to a double precision fixed-point number (4 bytes). ASCDF will only convert positive integer numbers. The ASCII string must contain eight characters and be right justified. Leading spaces are permitted and will be interpreted as zeros. The characters in the ASCII string are converted to BCD digits and restored in memory before the double fixed-point conversion occurs.

6.2.12 BCD to Binary Conversion - BCDBIN: Subroutine BCDBIN converts an eight-digit BCD number to a double precision fixed-point number. BCDBIN will only convert positive integer numbers. Each BCD digit occupies one byte. The eight-byte BCD number must be right justified. BCDBIN is called by subroutine ASCDF to complete the ASCII to a double precision fixed-point conversion.

7.0 REFERENCES.

1. 2648A Operating System Microcode, April 17, 1978, Hewlett-Packard No. 13255-90010.

2. 13290A/2649A Reference Manual, October 1977, Hewlett-Packard No. 13290-90003.

3. ISIS-II 8080/8085 Macro Assembler Operator's Manual, P/N 9800292C.

4. In-Circuit Emulator/80 Operator's Manual, P/N 98001850.

5. ICE-85 In-Circuit Emulator Operating Instructions for ISIS-II Users, P/N 98004638.

6. 8080/8085 Assembly Language Programming Manual, Intel Corporation Manual Order No. 98003010.

7. JHU/APL SDO 5949, Navigation Algorithm and Switchover Doctrine for Multiple-Option Loran Position Fixes, V. Schwab, March 1981.